МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования «МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт кибербезопасности и цифровых технологий

(наименование института, филиала)

КБ-8 «Информационное противоборство»

(наименование кафедры)

**КУРСОВАЯ РАБОТА**

по дисциплине Безопасность систем баз данных

(наименование дисциплины)

**Тема курсовой работы** Разработка и защита базы данных для автоматизированной информационной системы «Медицинское обследование в удаленном режиме»

**Студент группы** БББО-05-20 Кутьин Захар Сергеевич

(учебная группа, фамилия, имя, отчество студента) (подпись студента)

**Руководитель курсовой работы** доцент, к.т.н. Шукенбаев А.Б.

(должность, звание, ученая степень) (подпись

руководителя)

**Рецензент (при наличии)**

(должность, звание, ученая степень) (подпись рецензента)

**Работа представлена к защите** « » 20 г.

**Допущен к защите** « » 20 г.

МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования «МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт кибербезопасности и цифровых технологий

(наименование института, филиала)

КБ-8 «Информационное противоборство»

(наименование кафедры)

Утверждаю

Заведующий кафедрой КБ-8

Григорьев В.Р.

*(подпись) (ФИО)*

« » 20 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы по дисциплине**

**«Безопасность систем баз данных»**

**Обучающийся** Кутьин Захар Сергеевич  **Группа** БББО-05-20

**Тема** Разработка и защита базы данных для автоматизированной информационной системы «Медицинское обследование в удаленном режиме»

**Исходные данные:** База данных сотрудников транспортной компании

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

– разработка требований к базе данных автоматизированной информационной системы;

– проектирование подмоделей логического и физического уровня представления данных;

– реализация модели данных в СУБД SQL Server;

– построение системы защиты базы данных.

**Срок представления к защите курсового проекта (работы): до « » 20** г**. Задание на курсовой проект (работу) выдал (Шукенбаев А.Б.)**

***Подпись руководителя Ф.И.О. руководителя***

**« » 20 г*.***

**Задание на курсовую работу получил (Кутьин З.С.)**

***Подпись обучающегося Ф.И.О. исполнителя***

**Содержание**

[Введение 4](#_Toc136382973)

[Глава 1. Проектирование и реализация модели данных 8](#_Toc136382974)

[1.1 Системный анализ предметной области и анализ требований 8](#_Toc136382975)

[1.1.1. Анализ деятельности организации 8](#_Toc136382976)

[1.1.2. Категории пользователей 10](#_Toc136382977)

[1.1.3. Анализ требований 11](#_Toc136382978)

[1.1.4. Выбор и обоснование проектных решений по видам обеспечения 14](#_Toc136382979)

[1.1.5. Входная и выходная информация 16](#_Toc136382980)

[1.2 Проектирование базы данных 17](#_Toc136382981)

[1.2.1. Существующие в организации бизнес-процессы и процессы обработки данных исследуемого объекта 17](#_Toc136382982)

[1.2.2. Инфологическое моделирование 18](#_Toc136382983)

[1.2.3. Даталогическое проектирование 23](#_Toc136382984)

[1.3 Реализация 27](#_Toc136382985)

[1.3.1 Особенности реализации 27](#_Toc136382986)

[1.3.2 Тестирование и отладка хранимых процедур 28](#_Toc136382987)

[1.3.3 Тестирование и отладка триггеров 35](#_Toc136382988)

[1.4 Выводы по первой главе 39](#_Toc136382989)

[Глава 2. Построение системы защиты данных 40](#_Toc136382990)

[2.1 Создание регистрационных имен, имен пользователей и ролей 40](#_Toc136382991)

[2.2 Работа и реализация разрешений в базе данных 41](#_Toc136382992)

[2.3 Применение и реализация шифрования в базе данных 46](#_Toc136382993)

[2.4 Выводы по второй главе 55](#_Toc136382994)

[Заключение 57](#_Toc136382995)

[Список литературы 58](#_Toc136382996)

Введение

**Актуальность работы.** Данная курсовая работа посвящена разработке базы данных для хранения и обработки данных, полученных при проведении медицинского обследования в удаленном режиме сотрудников транспортной компании.

В настоящее время медицинские технологии развиваются с невероятной скоростью, и все больше людей обращаются за медицинской помощью в он­лайн-режиме. Наиболее актуальна тема дистанционного медицинского осмотра компаниям, в которых сотрудникам этой компании необходимо часто проходить медицинские обследования, что может тормозить бизнес-процессы. К данным организациям относятся транспортные компании.

Однако, для того чтобы обеспечить качественное медицинское обслужи­вание в удаленном режиме, необходимо иметь эффективную информацион­ную систему, которая бы обеспечивала хранение и обработку медицинских данных.

Разработка совершенной системы обработки и хранения данных о ре­зультатах медицинского осмотра, которая будет в полной мере обеспечивать потребности современных пользователей, является новшеством данного ис­следования.

**Целью данной курсовой работы** является проектирование, разработка и защита базы данных для хранения и обработки информации о результатах медицинского обследования в удаленном режиме сотрудников транспортной компании.

**Задачи** работы:

1. Изучить тонкости работы транспортной компании и компании, кото­рая предоставляет терминалы для обследования;
2. Найти способ хранить информацию о сотрудниках транспортной ком­пании, проходящих медицинское обследование в удаленном режиме;
3. Найти способ хранить и предоставлять информацию о результатах ме­дицинского обследования;
4. Сделать возможность изменения, внесения и удаления данных;
5. Сделать систему надежной и удобной для пользователя;
6. Обеспечить защиту системы.

**Объектом данного исследования** является информационная система для автоматизации проведения медицинских осмотров в удаленном режиме.

**Предметом исследования** выступает процесс разработки и реализации защищенной базы данных информационной системы для автоматизации про­ведения медицинских осмотров в удаленном режиме.

**Методы исследования:** изучение деятельности транспортных компа­ний и компаний, предоставляющих программное обеспечение и терминалы для проведения медицинского обследования в дистанционном режиме, озна­комление с документами и стандартами в области медицинских осмотров и создании медицинских изделий.

**Практическая значимость** курсовой работы заключается в том, что в результате ее выполнения будет создана функциональная и защищенная база данных, которая имеет важное значение для дальнейшего качественного раз­вития и усовершенствования систем медицинского обследования. Также дан­ная база данных сможет стать основой для внедрения или быть полностью ин­тегрирована в совместную деятельность транспортной компании и компании по предоставлению терминалов для медицинского обследования.

Для достижения поставленной цели использовались методы анализа и синтеза. Информационной базой исследования являются работы, опублико­ванные в сети Интернет, которые во многом посвящены теме изучения разра­ботки и защиты базы данных автоматизированной информационной системы «Медицинское обследование в удаленном режиме».

В структуру курсовой работы входят две главы.

В первой главе представлен анализ предметной области и разработка требований к создаваемой защищенной базе данных, в которую входят:

* категории пользователей, для которых создается система и их возмож­ности в базе данных;
* функциональные и нефункциональные требования;
* бизнес-требования;
* входные и выходные данные системы.

Описываются происходящие бизнес-процессы на предприятиях, также создаются инфологическая и даталогическая модели. Описываются компо­ненты входящие в эти модели: сущности и их взаимосвязи, отношения и атри­буты, поля и индексы.

Описывается реализация базы данных: особенности реализации, разра­ботка алгоритмов и кода хранимых процедур и триггеров, тестирование и от­ладка хранимых процедур и триггеров, разработка прототипа пользователь­ского интерфейса.

В конце главы указываются выводы.

Вторая глава посвящена защите базы данных от несанкционированного доступа, которая достигается путем использования регистрационных имен, имен пользователей, ролей и разрешений, а также шифрованием базы данных.

В конце главы формулируются выводы.

Создание базы данных для обмена данными между транспортной ком­панией и компанией по предоставлению медицинского обследования в удаленном режиме позволит повысить эффективность деятельности, а также время работы транспортной компании. Выполнение данной работы также позволит реализовать:

* предоставление систематического отчета об изменениях параметров здоровья водителя;
* обеспечение быстрого доступа к медицинским данным сотрудников, которые прошли медицинский осмотр;
* простой и быстрый поиск информации о пройденных медицинских осмотрах сотрудниками транспортной компании;
* доступность информации по каждому сотруднику, который прошел медицинский осмотр в удаленном режиме;
* обеспечение защиты базы данных от несанкционированного доступа.

1. Проектирование и реализация модели данных
2. Системный анализ предметной области и анализ требований

В главе 1 будет предоставлен системный анализ предметной области и анализ требований. Будут рассмотрены анализ деятельности организации, категории пользователей, для которых будет создана база данных, анализ требований, выбор и обоснование решений по видам обеспечения, а также входные и выходные данные.

1. Анализ деятельности организации

Необходимо создать базу данных для обработки, хранения и передачи информации между медицинским работником и сотрудником организации при прохождении им медицинского обследования в удаленном режиме, для чего необходимо изучить предметную область и соответствующие бизнес-процессы. Изучение происходит благодаря знакомству с документацией и государственными стандартами, регулирующими работу организации, а также все процессы происходящие внутри нее.

Компания по разработке аппаратно-программных средств для биомедицинских систем различного назначения, а также, созданию систем реального времени для регистрации, обработки, хранения и удаленной передачи медицинской информации предоставляет в аренду либо продает свои терминалы организациям для прохождения сотрудниками данной организации медицинского обследования в удаленном режиме.

Организация, сотрудник которой должен пройти медицинский осмотр в удаленном режиме, исходя из своих целей и задач, а также финансовых возможностей, арендует или покупает терминал для медицинского обследования в удаленном режиме.

Перед проведением медицинских осмотров руководитель организации регистрирует работника в системе автоматизации осмотров и оформляет согласие на обработку персональных данных.

Сотрудник организации при прохождении медицинского обследования в удаленном режиме самостоятельно выполняет «шаг за шагом» процедуру осмотра.

Система автоматически регистрирует результаты обследования аудио- и видеозаписи процедуры медицинского осмотра с последующей передачей информации о полученных результатах на мобильное устройство (смартфон, планшет) или на персональный компьютер как сотрудника организации, так и медработника.

Медработник подготавливает протокол о допуске к работе. Затем система передает сотруднику компании протокол медицинского заключения. После этого происходит распечатка на принтере 2 этикеток с результатами медосмотра и штампом о его прохождении, которые в свою очередь вклеиваются в путевой лист.

Бизнес-процессами компании по предоставлению медицинского обследования в удаленном режиме являются: составление договора с заказчиком, составление договора с медицинской организацией о предоставлении медицинских услуг, выдача результатов осмотра сотруднику, прошедшему медицинский осмотр.

Бизнес-процессами транспортной компании являются: регистрация водителя в информационной системе, выдача задания на рейс, проведение медицинского обследования водителя перед рейсом, обучение и инструктаж водителя (при необходимости).

В процессе медицинского осмотра в удаленном режиме сотрудника транспортной компании происходит хранение и обработка данных сотрудника, проходящего осмотр (ФИО, биометрические данные, состояние здоровья), а так же данные осмотра, включая дату, время, фото и видеозапись.

Транспортные компании предоставляют услуги перевозки людей и/или транспортировкой грузов. Сотрудники компании, которые проходят медицинское обследование в удаленном режиме являются водителями транспортных средств.

В результате проведенных анализа и исследования предметной области выдвигаются следующие требования к системе:

1. Хранение информации о персональных данных сотрудников, проходящих медицинское обследование в удаленном режиме;
2. Хранение и обработка информации о состоянии здоровья сотрудников, прошедших медицинское обследование в удаленном режиме;
3. Хранение информации о результатах медицинского обследования;
4. Хранение информации о медицинском работнике, проводившем обследование;
5. Категории пользователей

Разрабатываемая база данных предназначена для таких пользователей как:

* руководитель транспортной компании – регистрирует сотрудника своей организации (водителя или др.) в системе и оформляет согласие на обработку персональных данных. После прохождения сотрудником медицинского осмотра имеет право просмотреть результаты этого осмотра.
* сотрудник (водитель или др.) – проходя все этапы осмотра, вносит данные о состоянии своего здоровья, а после обследования получает его результаты.
* медицинский работник (врач-специалист) – по полученным данным от сотрудника, включая просмотр видеозаписи процедуры осмотра, проводит их анализ. После этого подготавливает протокол заключения о допуске к работе (рейсу).
* администратор сервера компании, предоставившей терминал для медицинского осмотра в удаленном режиме – следит за работоспособностью системы.

Руководитель транспортной компании не имеет доступа ко всей базе данных. Он может вносить, просматривать и обновлять данные о своих сотрудниках. А также по запросу получать результаты медицинского осмотра у конкретного сотрудника или у всех сотрудников, прошедших обследование в определенную дату.

Сотрудники, проходящие осмотр, имеют доступ вносить данные о себе (состояние здоровья), просматривать их и получать результаты медицинского обследования.

Медицинский работник имеет право просматривать данные о сотруднике и вносить данные о результатах обследования.

Администрирование базы данных осуществляет администратор сервера. Он имеет право вносить, обновлять и удалять данные о сотрудниках транспортной компании, данные о медицинском работнике. Важно отметить, только администратор имеет доступ к удалению данных из базы данных.

Все остальные сотрудники как транспортной компании, так и компании, предоставившей терминалы для медицинского обследования, не имеют никакого доступа к базе данных. К ним относится:

* сотрудники отдела кадров;
* сотрудники бухгалтерии;
* менеджеры;
* и др.

1. Анализ требований

В данной работе будет рассмотрена автоматизация медицинского обследования в удаленном режиме. Разрабатываемая база данных будет хранить данные о сотрудниках транспортной компании и данные медицинского специалиста.

Транспортная компания представляет собой полноценное рабочее предприятие, которая занимается перевозками людей и/или транспортировкой грузов.

Компания, которая предоставляет терминалы для обследования, занимается исследованиями, разработкой и производством программных и технических средств измерения, обработки, визуализации, хранения и дистанционной передачи данных для биомедицинских систем различного назначения.

В результате исследования и анализа предметной области была найдена потребность в создании базы данных для увеличения эффективности работы транспортной компании, а конкретно для ускорения процесса допуска водителя к рейсу. При анализе системы были сформулированы функциональные и нефункциональные требования к базе данных о результатах медицинского обследования в удаленном режиме сотрудником транспортной компании.

К функциональным требованиям можно отнести:

1. Хранение данных о медицинском обследовании водителей транспортной компании;
2. Возможность обновления и удаления существующих записей о водителях;
3. Защита данных о результатах медицинского обследования, а также личной информации водителей, проходящих медицинское обследование в удаленном режиме;
4. Высокое быстродействие системы;
5. Независимость данных;
6. Стандартизация построения и использования базы данных (СУБД);
7. Многопользовательский доступ к данным;
8. Интуитивно понятный интерфейс пользователя.

Независимость данных означает, что они не зависят от конкретного человека или системы, и могут быть использованы и доступны любому пользователю в соответствии с правилами доступа безопасности. Другими словами, данные могут быть использованы в разных контекстах без изменения своей структуры и формата. Независимость данных обеспечивает возможность обмена и совместимость между различными системами.

Стандартизация является важным аспектом обеспечения независимости данных. Она позволяет разным приложениям и системам работать с одной и той же базой данных, независимо от того, какая СУБД используется.

К нефункциональным требованиям к базе данных хранения и обработки данных о результатах медицинского обследования в удаленном режиме относится:

1. Создание рассматриваемой базы данных на базе программы MS SQL Server 2019;
2. Требуемый объем оперативной памяти не менее 2 Гб;
3. Скорость процессора, требуемый объем на жестком диске.

К требованиям по защите информации относится:

1. Реализация аутентификации и авторизации;
2. Разграничение прав доступа к данным;
3. Защита базы данных путём шифрования данных в ячейках и прозрачного шифрования;
4. Безопасность данных – защита данных от несанкционированного доступа, использования, изменения, раскрытия, уничтожения или повреждения.

Безопасность данных включает их целостность. Целостность данных – устойчивость хранимых данных к разрушению и уничтожению, связанных с неисправностями технических средств, системными ошибками и ошибочными действиями пользователей.

Целостность обеспечивается триггерами целостности – специальными приложениями-программами, работающими при определенных условиях. Защита данных от несанкционированного доступа предполагает ограничение доступа к конфиденциальным данным и может достигаться:

* введением системы паролей;
* получением разрешений от администратора базы данных (АБД);
* запретом от АБД на доступ к данным;
* формированием видов – таблиц, производных от исходных и предназначенных конкретным пользователям.

1. Выбор и обоснование проектных решений по видам обеспечения

Для выбора подходящей СУБД, где будет создана база данных, необходимо провести исследование и анализ существующих систем управления и выбрать наиболее удобную и выгодную предприятию систему.

MS SQL Server – это система управления реляционными базами данных, разработанная компанией Microsoft. Она обладает высокой производительностью и отказоустойчивостью, тесно интегрирована с ОС Windows. Этот сервер поддерживает удаленные подключения, работает с многими популярными типами данных, дает возможность создавать триггеры и хранимые процедуры, имеет практичные и удобные утилиты для настройки.

В качестве основного языка запросов в MS SQL Server используется Transact-SQL (совместная разработка Microsoft и Sybase). При этом Transact-SQL — это реализация стандарта ANSI/ISO по SQL (структурированному языку запросов), но имеющая некоторые расширения. Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API (например, MS SQL Server Management Studio). СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Сегодня СУБД MS SQL широко применяется при работе с базами данных (БД) самых разных размеров, начиная от персональных, заканчивая крупными БД масштаба предприятия.

MS SQL Server долгое время был доступен исключительно для Windows, однако начиная с версии 16 эта система доступна и на Linux. Кроме того, MS SQL Server позволяет установить разные версии на одном компьютере.

SQL Server характеризуется следующими особенностями:

* Высокая производительность – MS SQL Server работает достаточно быстро.
* Надежность и безопасность – MS SQL Server предоставляет шифрование данных.
* С MS SQL Server относительно легко работать и вести администрирование.

Проанализировав возможности и особенности разных СУБД, было решено, что база данных будет создана в среде MS SQL Server, т.к. именно эта СУБД в данном случае отвечает всем требованиям и имеет меньше всего недостатков.

При проектировании базы данных будет использована онлайн-платформа **draw.io,** т.к. она имеет все необходимые для составления диаграмм функции и при этом не требует установки – можно работать в режиме онлайн.

Также для создания диаграмм, в которых описываются бизнес-процессы организации, может быть использовано инструментальное CASE-средство CA ERWin Process Modeler.

CA ERWin Process Modeler – CASE-средство для моделирования деловых процессов, позволяющая создавать диаграммы в нотациях IDEF0 (функционального моделирования), IDEF3 (описания процессов), DFD (моделирования потоков данных). В процессе моделирования CA ERWin Process Modeler позволяет переключиться с нотации IDEF0 на любой ветви модели на нотацию IDEF3 или DFD и создать смешанную модель. CA ERWin Process Modeler поддерживает функционально-стоимостной анализ (ABC).

1. Входная и выходная информация

Входными данными для базы данных являются данные о состоянии здоровья и медицинских показателях сотрудников, проходящих медицинское обследование в удаленном режиме, а также другие данные сотрудника транспортной компании, которые вносит руководитель этой компании путем создания записей о сотрудниках в системе (ФИО, контактные данные, возраст, пол и др.).

Данные о состоянии здоровья и медицинских показателях водителей вводятся в систему каждый рейс – это примерно 1 раз в 2 или 3 дня (при этом могут вводится как чаще, так и реже). ФИО, контактные и другие данные сотрудников компании, проходящих обследование, вводятся в базу данных при первом обращении к организации предоставляющей терминалы для медицинского обследования в удаленном режиме и обновляются не чаще 1 раза в полгода (для данных о сотрудниках, чаще при обновлении каких-либо данных о них).

Выходными данными является протокол медицинского обследования – дата прохождения обследования и результаты медицинского осмотра. Выходными данными могут пользоваться сотрудники, проходившие медицинский осмотр, медицинский работник, проводивший обследования, а также руководитель компании, в которой работают сотрудники, для контроля за состоянием здоровья своих работников.

1. Проектирование базы данных

В главе 2 представлено проектирование базы данных перед ее реализацией. В ней будут рассмотрены существующие в транспортной компании бизнес-процессы и процессы обработки исследуемого объекта, а также инфологическая и даталогическая модели и их составляющие (сущности и их взаимосвязи, отношения и атрибуты, поля и индексы).

1. Существующие в организации бизнес-процессы и процессы обработки данных исследуемого объекта

Основными бизнес-процессами транспортной компании являются: прием и обработка заказов, составление маршрута и планирование доставки, управление персоналом, обслуживание клиентов.

Контекстная диаграмма, которая отображает основную функцию системы, а так же входные и выходные данные, управляющую и регламентирующую функции, представлена в приложении рисунке 1. Модель построена с использованием CA ERWin Process Modeler.

Входными данными будут данные о заказе на перевозку грузов или людей от клиентов, данные о грузах, данные об адресе получателя, информация о стоимости доставки.

Выходными данными являются договор об оказании услуг, счет на оплату.

Управляющими данными являются государственные стандарты (ГОСТы) на составление договора об оказании услуг, федеральные законы, регулирующие деятельность транспортных компаний, федеральный закон от 29.12.2022 № 629-ФЗ.

Механизмом исполнения являются руководитель организации, менеджеры и водители, медицинский работник, транспортные средства, а также различная техника и оборудование.

В приложении на рисунке 2 показана диаграмма декомпозиции первого уровня, которая является детализацией контекстной диаграммы. Контекстная диаграмма является родительской по отношению к диаграмме декомпозиции, которая, в свою очередь, является дочерней для контекстной диаграммы. Дочерняя диаграмма более подробно описывает функции системы, которые реализуются в рамках основной функции, отображенной в родительской диаграмме.

В приложении на рисунке 3 показана диаграмма декомпозиции второго уровня, где подробно описан бизнес-процесс «Управление персоналом». Именно в данном процессе будет задействована создаваемая в данной курсовой работе база данных о медицинском обследовании в удаленном режиме.

1. Инфологическое моделирование

В этом разделе будет описано создание инфологической (концептуальной) модели. Она нужна для описания структуры данных, которые будут храниться в базе данных. При этом помогает представить предметную область на более понятном для восприятия человеком и независимой от какой-либо СУБД форме. Концептуальная модель представляется в графическом виде.

Сущность – собирательное понятие, некоторая абстракция реально существующего объекта, процесса или явления, о котором необходимо хранить информацию в системе.

Каждая сущность имеет атрибуты – поименованные характеристики сущности, принимаемые значения некоторого множества значений.

1. Сущность «Сотрудник».

Сотрудниками являются водители, проходящие медицинский осмотр в удаленном режиме. Водителями являются физические лица.

Атрибуты сущности «Сотрудник»:

* ID сотрудника – первичный ключ; идентификационный номер, уникальный для каждого сотрудника, который является номером и серией паспорта;
* Фамилия;
* Имя;
* Отчество;
* Возраст;
* Пол;
* Телефон – номер телефона для обратной связи с сотрудником;
* Электронная почта – электронный адрес также необходимый для обратной связи с сотрудником.

2. Сущность «Паспорт».

Сущность «Паспорт» включает в себя информацию о паспорте сотрудника, который проходит медицинский осмотр в удаленном режиме.

Атрибуты сущности «Паспорт»:

* Серия и номер – первичный ключ; серия и номер паспорта уникальны для каждого сотрудника;
* Дата выдачи – дата выдачи паспорта;
* Кем выдан – информация о государственном органе, который выдал паспорт сотруднику;
* Код подразделения – уникальный номер, присваиваемым каждому отделу управления по вопросам миграции;

3. Сущность «Врач».

Врачами являются сотрудники медицинской организации, с которой компания, предоставляющая на продажу или аренду терминалы для медицинского обслуживания в удаленном режиме, заключает договор об оказании медицинских услуг.

Атрибуты сущности «Врач»:

* ID врача – первичный ключ; идентификационный номер, уникальный для каждого врача;
* Фамилия;
* Имя;
* Отчество;
* Специализация – специализация врача, который проводит медицинское обследование;
* Электронная почта – электронный адрес медицинской организации или врача, проводившего осмотр, необходимый для связи с врачом.

4. Сущность «Медицинский осмотр».

Сущность «Медицинский осмотр» включает в себя медицинские показатели состояния водителя, который прошел медицинское обследование в удаленном режиме.

Атрибуты сущности «Медицинский осмотр»:

* ID осмотра – первичный ключ; идентификационный номер, уникальный для каждого медицинского осмотра;
* ID сотрудника;
* ID врача;
* Дата и время осмотра;

5. Сущность «Результаты медицинского осмотра».

Сущность «Результаты медицинского осмотра» включает в себя сведения о результатах осмотра: температура тела, артериальное давление(систолическое и диастолическое), пульс, концентрация алкоголя в выдыхаемом воздухе, жалобы сотрудника на самочувствие. Результаты осмотра доступны сотруднику, проходившему осмотр, а также руководителю организации, в которой он работает.

Атрибуты сущности «Результаты медицинского осмотра»:

* ID результата;
* Температура тела;
* Систолическое давление;
* Диастолическое давление;
* Пульс;
* Концентрация алкоголя в выдыхаемом воздухе;
* Жалобы – жалобы сотрудника на свое самочувствие.

6. Сущность «Протокол».

Сущностью «Протокол» является документ, который формируется по итогам медицинского осмотра. Содержит медицинские показатели сотрудника.

Атрибуты сущности «Протокол»:

* Номер протокола – первичный ключ; идентификационный номер, уникальный для каждого протокола;
* Заключение врача – итог анализа медицинским работником медицинских показателей сотрудника.

Сущности состоят в определенных отношениях, средством представления которых выступают связи. Эти связи можно описать определенным глаголом с заданной в имеющейся взаимной зависимости между сущностями модальностью («должен» и «может»). В данной предметной области среди выделенных сущностей существуют следующие связи:

Связь «Сотрудник» – «Паспорт» (1:1).

У каждого сотрудника может быть только один паспорт и у каждого паспорта может быть только один владелец.

Связь «Сотрудник» – «Медицинский осмотр» (1:М).

У каждого сотрудника может быть несколько медицинских осмотров, но каждый осмотр относится только к одному сотруднику.

Связь «Врач» – «Медицинский осмотр» (1:М).

У каждого врача может быть несколько медицинских осмотров, но каждый осмотр проводится только одним врачом.

Связь «Медицинский осмотр» – «Результаты медицинского осмотра» (1:1).

У каждого медицинского осмотра может быть только один результат; каждый результат относится только к одному осмотру.

Связь «Результаты медицинского осмотра» — «Протокол» (1:1).

По каждому результату медицинского осмотра может быть составлен только один протокол и каждый протокол соответствует только одному результату медицинского осмотра.

В предыдущих разделах был проведен анализ предметной области, также были выделены сущности и связи, существующие между ними. Инфологическая модель была создана на основе исследований и анализа необходимых сущностей и связей между ними в предметной области с помощью онлайн-системы draw.io.

Инфологическая модель базы данных автоматизированной информационной системы медицинского обследования в удаленном режиме представлена на рисунке 1.1.

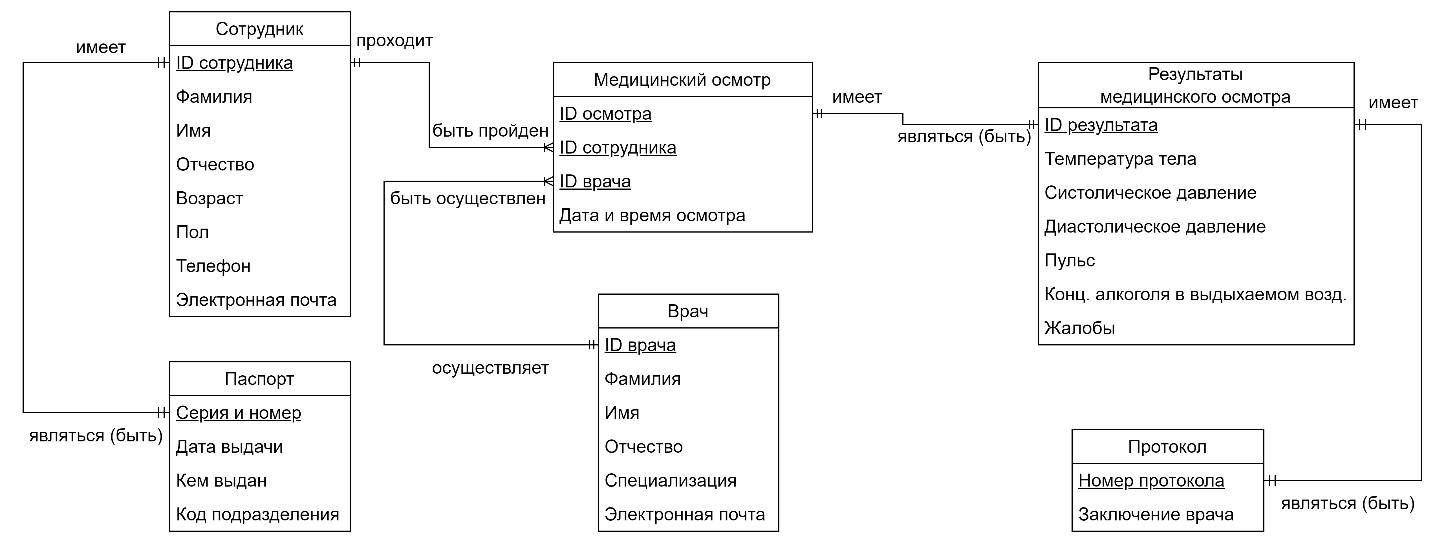


Рисунок 1.1 – Инфологическая модель (draw.io)

1. Даталогическое проектирование

В этом разделе будет описано создание даталогической модели системы. Даталогическая модель базы данных - это модель, которая описывает структуру данных, хранимых в базе данных, и связи между ними. Она определяет, какие данные будут храниться в базе данных, их качественные и количественные характеристики, как они будут организованы и как будут связаны друг с другом. Другими словами, даталогическая модель определяет логическую структуру базы данных.

Логическая структура и заполненная база данных являются отображением реальной исследуемой предметной области, поэтому на выбор многих проектных решений оказывает влияние особенности предметной области, которая отражена в инфологической модели.

Даталогическая модель представлена в графическом виде.

Далее будут описаны отношения и атрибуты, которые войдут в даталогическую модель. Свойства атрибутов отношений представлены в приложении в таблице 1.

1. Отношение «Employee».

Отношение «Employee» соответствует сущности «Сотрудник» и атрибуты этого отношения соответствуют атрибутам сущности:

* emp\_id – ID сотрудника, первичный ключ; идентификационный номер, уникальный для каждого сотрудника, который соответствует номеру и серии паспорта;
* emp\_fname – фамилия сотрудника;
* emp\_name – имя сотрудника;
* emp\_surname – отчество сотрудника;
* emp\_age – возраст сотрудника;
* emp\_gender – пол сотрудника;
* emp\_phone – номер телефона сотрудника;
* emp\_email – адрес электронной почты сотрудника.

2. Отношение «Passport».

Отношение «Passport» соответствует сущности «Паспорт» и атрибуты этого отношения соответствуют атрибутам сущности:

* pass\_id – номер и серия паспорта;
* pass\_date – дата выдачи паспорта;
* pass\_issued – информация об отделе управления по вопросам миграции;
* pass\_code – код подразделения.

3. Отношение «Doctor».

Отношение «Doctor» соответствует сущности «Врач» и атрибуты этого отношения соответствуют атрибутам сущности:

* doc\_id – ID врача, первичный ключ; идентификационный номер, уникальный для каждого врача;
* doc\_fname – фамилия врача;
* doc\_name – имя врача;
* doc\_surname – отчество врача;
* doc\_spec – специализация врача;
* doc\_email – электронный адрес медицинской организации или врача.

4. Отношение «MedСheckup».

Отношение «MedCheckup» соответствует сущности «Медицинский осмотр» и атрибуты этого отношения соответствуют атрибутам сущности:

* check\_id – ID осмотра, первичный ключ; идентификационный номер, уникальный для каждого медицинского осмотра;
* check\_date – дата и время осмотра;

5. Отношение «Results».

Отношение «Results» соответствует сущности «Результаты медицинского осмотра» и атрибуты этого отношения соответствуют атрибутам сущности:

* res\_id – первичный ключ; соответствует ID медицинского осмотра.
* res\_temp – температура тела;
* res\_sys\_pres – систолическое давление;
* res\_dias\_pres – диастолическое давление;
* res\_pulse – пульс;
* res\_alco – концентрация алкоголя в выдыхаемом воздухе;
* res\_compl – жалобы на самочувствие.

6. Отношение «Protocol».

Отношение «Protocol» соответствует сущности «Протокол» и атрибуты этого отношения соответствуют атрибутам сущности:

* prot\_num – номер протокола, первичный ключ; идентификационный номер, уникальный для каждого протокола, соответствует ID результата медицинского осмотра;
* prot\_concl – заключение врача.

Для дальнейшей разработки даталогической модели необходимо рассмотреть несколько важных определений.

Ключ сущности – это атрибут или набор атрибутов, который уникально идентифицирует каждую сущность. При этом сущность может иметь несколько различных ключей.

Первичный ключ – атрибут или набор атрибутов, однозначно идентифицирующие экземпляр сущности.

Внешний ключ – это набор атрибутов отношения, содержащий ссылки на ключи другого или того же отношения.

Индексы в базах данных - это структуры данных, которые позволяют быстро получать доступ к одной или нескольким строкам данных. Использование индексов позволяет ускорить выполнение запросов к базе данных, так как при поиске нужных записей не нужно проходить по всей таблице, тем самым обеспечивается быстрый доступ к данным.

Все ключи и индексы отношений описаны в приложении в таблице 2.

После исследования всех отношений и связей между ними с помощью онлайн-системы draw.io была построена даталогическая модель.

Даталогическая модель базы данных автоматизированной информационной системы медицинского обследования в удаленном режиме представлена на рисунке 1.2.

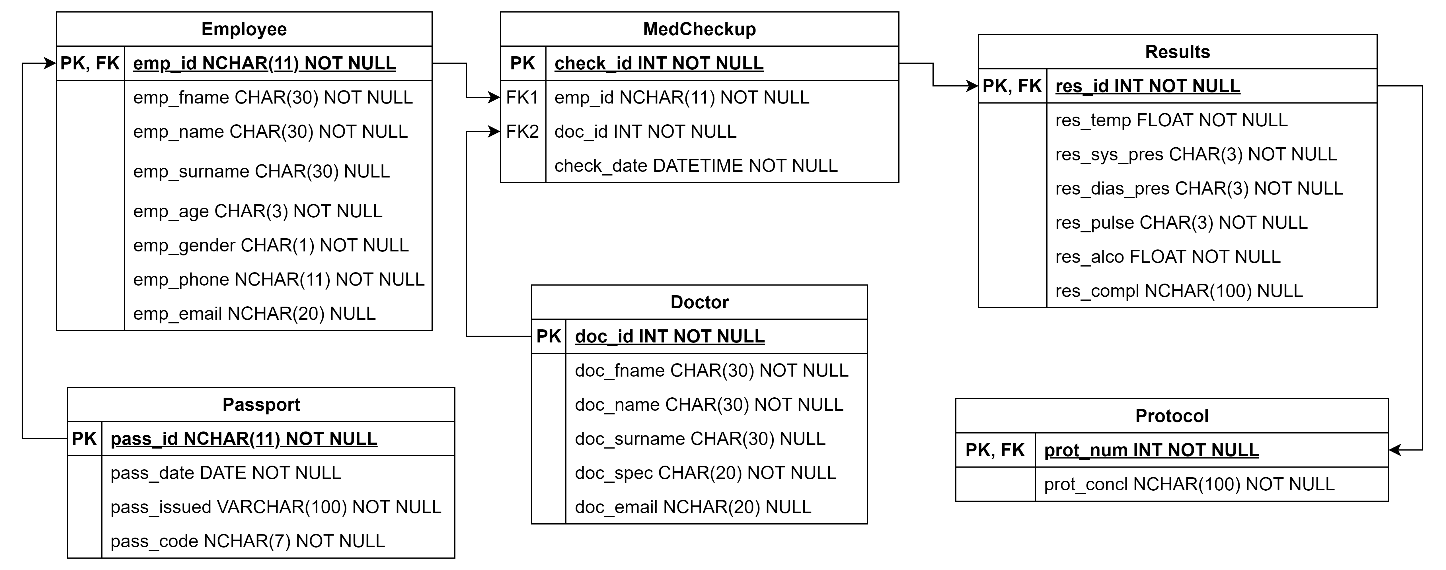


Рисунок 1.2 - Даталогическая модель (draw.io)

1. Реализация

В данном параграфе будет описана реализация спроектированной базы данных, а также различные особенности реализации. Здесь же будут рассмотрены разработка хранимых процедур и триггеров, их тестирование и отладка.

1. Особенности реализации

База данных будет создана в выбранной в ходе исследования и анализа СУБД Microsoft SQL Server. База данных будет называться «MedExamSys».

В базе данных «MedExamSys» 6 таблиц, которые соответствуют рассмотренным ранее во второй части отношениям:

1) Employee;

2) Passport;

3) Doctor;

4) MedСheckup;

5) Results;

6) Protocol.

В таблицах указываются первичные и внешние ключи, создаются необходимые индексы, описанные ранее. Также создаются связи между отношениями (с помощью внешних ключей).

Коды создания для каждой таблицы и связи между ними показаны в приложении на рисунках 4-9. В приложении на рисунках 10-15 соответственно заполненные данными таблицы. Диаграмма базы данных продемонстрирована в приложении на рисунке 16.

При реализации поле emp\_gender (пол сотрудника, который проходит медицинский осмотр) отношения Employee следует сделать полем со списком, в который входят: «М», «Ж». При этом список будет являться ограниченным из двух вариантов.

1. Тестирование и отладка хранимых процедур

Хранимая процедура — это набор SQL-инструкций, написанных на Transact-SQL который компилируется один раз и хранится на сервере. Хранимые процедуры используются для упрощения и оптимизации работы с базой данных. Они позволяют избежать повторения одних и тех же запросов, ускорить выполнение сложных операций и обеспечить безопасность данных.

В этом разделе будет описана разработка хранимых процедур, которые обеспечивают улучшенную работу созданной базы данных.

Первая хранимая процедура: Select\_concl\_for\_date.

Эта процедура выдает информацию о всех заключениях врачей в протоколах по результатам прохождения сотрудниками медицинских осмотров в конкретную дату осмотра.

Входным параметром является @med\_date типа DATE – дата проведения медицинских обследований.

Процедура выполняет запрос SELECT для вывода данных из таблиц Employee, MedCheckup и Protocol, а именно фамилий, имен и отчеств сотрудников, также заключения врачей и дату медицинского осмотра.

Информация будет выведена только при условии, что входной параметр @med\_date будет равен одному из значений столбца check\_data таблицы MedCheckup, ID сотрудника из столбца emp\_id таблицы Employee будет совпадать с ID сотрудника из столбца emp\_id таблицы MedCheckup, а также ID медицинского осмотра из столбца check\_id таблицы MedCheckup будет равен номеру протокола из столбца prot\_num таблицы Protocol.

Код создания данной процедуры показан в приложении на рисунке 17. Пример работы процедуры показан в приложении на рисунке 18.

Вторая хранимая процедура: Select\_res\_for\_emp.

Эта процедура выдает информацию о результатах медицинского осмотра, а именно показателей состояния здоровья сотрудника по его фамилии, имени и отчеству.

Входными параметрами процедуры являются @emp\_fname CHAR(30) – фамилия сотрудника, @emp\_name CHAR(30) – имя сотрудника, @emp\_surname CHAR(30) – отчество сотрудника, @flag BIT – параметр по умолчанию, необходим для использования процедуры в триггере Insert\_prot\_concl.

Процедура выполняет запрос SELECT для вывода данных из таблиц MedCheckup, Results, если вызов процедуры осуществляется вне триггера Insert\_prot\_concl, а именно ID медицинского осмотра, даты медицинского осмотра, температуры, систолического и диастолического давления, пульса, концентрации алкоголя в выдыхаемом воздухе, а также жалоб сотрудника на самочувствие. Если же вызов процедуры осуществляется в триггере Insert\_prot\_concl, то вывод данных осуществляется из таблиц MedCheckup, Results и Protocol, а именно ID медицинского осмотра, даты медицинского осмотра, температуры, систолического и диастолического давления, пульса, концентрации алкоголя в выдыхаемом воздухе, жалоб сотрудника на самочувствие, а также заключения врача из протокола.

Вначале в процедуре проверятся условие: @flag = 0. Если условие истинно, это значит что вызов процедуры осуществляется вне триггера Insert\_prot\_concl. После этого отрабатывает запрос SELECT на вывод информации описанной выше только при условии, что значение столбца emp\_fname из таблицы Employee будет равен входному параметру @emp\_fname, значение столбца emp\_name из таблицы Employee будет равно входному параметру @emp\_name, значение столбца emp\_surname из таблицы Employee будет равно входному параметру @emp\_surname, значение столбца emp\_id из таблицы Employee будет равно значению столбца emp\_id из таблицы MedCheckup, значение столбца check\_id из таблицы MedCheckup будет равно значению столбца res\_id из таблицы Results. Если условие ложно, это значит вызов процедуры осуществляется внутри триггера Insert\_prot\_concl. После отрабатывает аналогичный выше написанному запрос SELECT на вывод данных. Отличием от предыдущего будет вывод информации о заключении врача по результатам медицинского осмотра, а также в условие вывода добавляется условие, что значение столбца res\_id из таблицы Results должно совпадать со значением столбца prot\_num из таблицы Prototcol.

Код создания данной процедуры показан в приложении на рисунке 19. Пример работы процедуры показан в приложении на рисунке 20.

Третья хранимая процедура: Select\_emp\_ready.

Эта процедура выдает информацию о сотрудниках, которые по результатам медицинского осмотра имеют заключения врача «Признаки состояний и заболеваний отсутствуют».

Процедура выполняет запрос SELECT для вывода данных из таблиц Employee и MedCheckup, а именно фамилии, имени и отчества сотрудника, а также даты проведения медицинского осмотра.

Информация будет выведена только при условии, что значение столбца emp\_id из таблицы Employee будет равно значению столбца emp\_id из таблицы MedCheckup, значение столбца check\_id таблицы MedCheckup будет равно значению столбца prot\_num из таблицы Protocol, а также значение столбца prot\_concl из таблицы Protocol должен быть равен следующему значению – «Признаки состояний и заболеваний отсутствуют».

Код создания данной процедуры показан в приложении на рисунке 21. Пример работы процедуры показан в приложении на рисунке 22.

Четвертая хранимая процедура: Insert\_med\_check.

Эта процедура добавляет в таблицу MedCheckup новую строку с данными о новом медицинском осмотре. Если ID нового осмотра уже существует, то старые данные обновятся новыми.

Входными параметрами являются @new\_check\_id INT – ID нового медицинского осмотра, @emp\_id NCHAR(11) – ID сотрудника, @doc\_id INT – ID доктора, @new\_check\_date DATETIME – дата проведения нового медицинского осмотра.

Вначале процедуры проверятся условие с помощью оператора EXISTS, если значение параметра @new\_check\_id уже существует в таблице MedCheckup, то выполняется обновление данных таблицы MedCheckup с помощью оператора UPDATE, а именно столбцы emp\_id, doc\_id и check\_data заполняются соответственно значениями входных параметров @emp\_id, @doc\_id и @new\_check\_date, где значение столбца check\_id из таблицы MedCheckup будет совпадать с входным параметром @new\_check\_id.

Если условие вначале процедуры оказывается ложно, то будет выполнен оператор INSERT, который вставит новую строку данных в таблицу MedCheckup со значениями входных параметров.

Код создания данной процедуры показан в приложении на рисунке 23. Пример работы процедуры показан в приложении на рисунке 24.

Пятая хранимая процедура: Update\_data\_pass.

Эта процедура обновляет данные строки по всем столбцам таблицы Passport по номеру и серии паспорта.

Входными параметрами процедуры являются @new\_pass\_id NCHAR(11) – номер и серия паспорта, данные которого необходимо обновить, @new\_pass\_date DATE – новая дата выдачи паспорта, @new\_pass\_issued VARCHAR(100) – новая информация об отделе УФМС, выдавшем паспорт, @new\_pass\_code NCHAR(7) – новый код подразделения.

После объявления входных параметров с помощью оператора DISABLE отключается триггер Update\_old\_pass для корректной работы данной процедуры. После этого проверяется условие с помощью оператора EXISTS, если значение параметра @new\_pass\_id существует в таблице Passport, то выполняется обновление данных в таблице Passport с помощью оператора UPDATE, а именно столбцы pass\_date, pass\_issued и pass\_code заполняются соответственно значениями входных параметров @new\_pass\_date, @new\_pass\_issued и @new\_pass\_code.

Если условие в начале процедуры ложно, то с помощью оператора RAISERROR создается сообщение об ошибке с текстом «Таких номера и серии паспорта не существует для обновления».

Код создания данной процедуры показан в приложении на рисунке 25. Пример работы процедуры показан в приложении на рисунках 26-27.

Шестая хранимая процедура: Update\_pass\_id.

Эта процедура обновляет значение столбца pass\_id (номер и серия паспорта) таблицы Passport, заменяя старые значения на новые.

Входными параметрами процедуры являются @old\_pass\_id NCHAR(11) – старые значения номера и серии паспорта, существующих в таблице Passport, @new\_pass\_id NCHAR(11) – новые значения номера и серии паспорта, которые заменят старые.

После объявления входных параметров с помощью оператора ENABLE включается триггер Update\_old\_pass для корректной работы данной процедуры. После этого проверяется условие с помощью операторов NOT и EXISTS, если значение параметра @new\_pass\_id не существует в таблице Passport, то выполняется обновление данных в таблице Passport с помощью оператора UPDATE, а именно столбец pass\_id заполняется значением входного параметра @new\_pass\_id, где значение столбца pass\_id таблицы Passport совпадает со значением входного параметра @old\_pass\_id.

Если условие в начале процедуры ложно, то с помощью оператора RAISERROR создается сообщение об ошибке с текстом «Такие номер и серия паспорта уже существуют в базе данных».

Важно отметить, что при обновлении pass\_id, значение внешних ключей foreign\_pass таблицы Employee и foreign\_emp таблицы MedCheckup обновятся автоматически, так как при создании данных таблиц было применено предложение ON UPDATE CASCADE, которое позволяет выполнить каскадное обновление таблиц.

Код создания данной процедуры показан в приложении на рисунке 28. Пример работы процедуры показан в приложении на рисунках 29-30.

Седьмая хранимая процедура: Calc\_emp\_ready.

Эта процедура считает количество сотрудников, в заключении по результатам медицинского осмотра которых стоит «Признаки состояний и заболеваний отсутствуют».

Процедура выполняет запрос SELECT, в котором с помощью оператора COUNT() производится подсчет уникальных ID сотрудников, что достигается с помощью оператора DISTINCT, в заключении которых стоит «Признаки состояний и заболеваний отсутствуют».

Подсчет производится только тех ID сотрудников, где значение столбца check\_id в таблице MedCheckup равно значению столбца prot\_num таблицы Protocol и значение столбца prot\_concl равно «Признаки состояний и заболеваний отсутствуют».

Код создания данной процедуры показан в приложении на рисунке 31. Пример работы процедуры показан в приложении на рисунке 32.

Восьмая хранимая процедура: Delete\_emp.

Эта процедура удаляет сотрудника из базы данных по его фамилии, и имени и отчеству.

Входными параметрами являются @emp\_fname CHAR(30), @emp\_name CHAR(30), @emp\_surname CHAR(30) – фамилия, имя и отчество сотрудника, который будет удален.

В процедуре выполняется запрос DELETE, который удаляет строку таблицы Employee, где значение столбца emp\_fname таблицы Employee равно входному параметру @emp\_fname, значение столбца emp\_name таблицы Employee равно входному параметру @emp\_name и значение столбца emp\_surname таблицы Employee равно входному параметру @emp\_surname.

Важно отметить, что строки таблиц MedCheckup, Resultsl и Protocol, которые связаны с сотрудником, который был удален, также будут удалены, так как при создании данных таблиц было применено предложение ON DELETE CASCADE, которое позволяет выполнить каскадное удаление таблиц.

Код создания данной процедуры показан в приложении на рисунке 33. Пример работы процедуры показан в приложении на рисунке 34.

1. Тестирование и отладка триггеров

Триггер – это специальный механизм, который автоматически выполняется при определенных событиях или изменениях данных в таблицах. Каждый триггер состоит из имени, действия и исполнения.

В этом разделе будет описана разработка триггеров, которые обеспечивают улучшенную работу созданной базы данных.

Первый триггер: Update\_old\_pass.

Данный триггер срабатывает после обновления таблицы Passport.

Триггер с помощью оператора UPDATE обновляет данные таблицы Passport, а с помощью оператора SET выставляет столбцам pass\_code, pass\_date и pass\_issued значения соответственно '000-000', '1900-01-01' и 'Данные устарели', где значение столбца pass\_id таблицы Passport равно значению столбца pass\_id таблицы inserted.

Код создания данного триггера показан в приложении на рисунке 35. Пример выполнения триггера показан в приложении на рисунке 36.

Второй триггер: Insert\_prot\_concl.

Данный триггер срабатывает после добавления новых или обновлении старых значений таблицы Protocol.

Вначале триггера с помощью оператора DECLARE объявляются переменные, необходимые для его работы @new\_prot\_id INT – номер нового протокола, @emp\_fname CHAR(30) @emp\_name CHAR(30) и @emp\_surname CHAR(30) соответственно фамилия, имя и отчество сотрудника, протокол с заключением врача которого был добавлен или обновлен.

После объявления с помощью операторов SELECT переменные @new\_prot\_id, @emp\_fname, @emp\_name и @emp\_surname заполняются соответственно значениями столбца prot\_num таблицы inserted, столбцов emp\_fname, emp\_name и emp\_surname таблицы Employee, где значение столбца emp\_id таблицы Employee равно значению столбца emp\_id таблицы MedCheckup и значение столбца check\_id таблицы MedCheckup равно значению переменной @new\_prot\_id.

После чего запускается хранимая процедура Select\_res\_for\_emp с входными параметрами @emp\_fname, @emp\_name, @emp\_surname, 1.

Код создания данного триггера показан в приложении на рисунке 37. Пример выполнения триггера показан в приложении на рисунке 38.

Третий триггер: Check\_date.

Данный триггер срабатывает после добавления новых или обновлении старых значений таблицы MedCheckup.

Вначале триггера с помощью оператора DECLARE объявляются переменные, необходимые для его работы @med\_check\_id INT – ID медицинского осмотра, @check\_date DATE – дата проведения осмотра.

После объявления с помощью операторов SET переменные @med\_check\_id и @check\_date заполняются соответственно максимальным значением столбца check\_id из таблицы MedCheckup, значением столбца check\_data таблицы MedCheckup, где значение столбца check\_id таблицы MedCheckup равно значению переменной @med\_check\_id.

Далее проверяется условие, если значение переменной @check\_date больше сегодняшней даты, значение которой возвращает функция GETDATE(), то с помощью оператора RAISERROR создается сообщение об ошибке с текстом «Дата прохождения мед. осмотра не может быть позже сегодняшней даты!».

Код создания данного триггера показан в приложении на рисунке 39. Пример выполнения триггера показан в приложении на рисунке 40.

Четвертый триггер: Check\_pass\_id.

Данный триггер срабатывает после добавления новых или обновлении старых значений таблицы Passport.

Вначале триггера с помощью оператора EXISTS проверяется условие, если значение столбца pass\_id таблицы inserted состоит из вначале четырех цифр, пробела и шести цифр в конце, то триггер завершает работу.

Если условие оказывается ложно, то с помощью оператора RAISERROR создается сообщение об ошибке с текстом «Ошибка при вводе данных паспорта».

Код создания данного триггера показан в приложении на рисунке 41. Пример выполнения триггера показан в приложении на рисунке 42.

Пятый триггер: Check\_phone.

Данный триггер срабатывает после добавления новых или обновлении старых значений таблицы Employee.

Вначале триггера с помощью оператора DECLARE объявляется переменная, необходимая для его работы, @phone NCHAR(11) – номер телефона сотрудника.

После объявления с помощью оператора SELECT переменная @phone заполняется значением столбца emp\_phone таблицы inserted.

Затем проверяется условие, если значение переменной @phone состоит из вначале цифры восемь и десяти за ней любых цифр, то триггер завершает работу.

Если условие не выполняется, то с помощью оператора RAISERROR создается сообщение об ошибке с текстом «Некорректный номер телефона (ожидается ввод только цифр) или номер начинается не с цифры ''8''!».

Код создания данного триггера показан в приложении на рисунке 43. Пример выполнения триггера показан в приложении на рисунке 44.

Шестой триггер: Check\_emp\_name.

Данный триггер вызывается после добавления новых или обновлении старых значений таблицы Employee.

В триггере с помощью трех операторов EXISTS проверяется условие, если значения столбцов emp\_fname, emp\_name и emp\_surname из таблицы inserted содержат хотя бы одну цифру, то с помощью оператора RAISERROR создается сообщение об ошибке с текстом «ФИО сотрудника не должно содержать цифры».

Код создания данного триггера показан в приложении на рисунке 45. Пример выполнения триггера показан в приложении на рисунке 46.

Седьмой триггер: Check\_doc\_name.

Работа данного триггера аналогична работе триггера Check\_emp\_name, за исключением того, что триггер срабатывает после добавления новых или обновлении старых значений таблицы Doctor и для условия берутся значения столбцов doc\_fname, doc\_name и doc\_surname.

Код создания данного триггера показан в приложении на рисунке 47. Пример выполнения триггера показан в приложении на рисунке 48.

Восьмой триггер: Delete\_pass.

Данный триггер срабатывает после удаления значений из таблицы Empployee.

В триггере с помощью оператора DELETE из таблицы Passport удаляется строка с данными, где значение столбца pass\_id таблицы Passport равно значению столбца emp\_id таблицы deleted.

Код создания данного триггера показан в приложении на рисунке 49. Пример выполнения триггера показан в приложении на рисунке 50.

1. Выводы по первой главе

В результате работы получилось создать базу данных для обеспечения хранения и обработки данных, полученных при проведении медицинского обследования в удаленном режиме сотрудников транспортной компании.

База данных содержит информацию о сотрудниках компании, а именно паспортные данные, фамилию, имя и отчество. Медицинские показатели о состоянии здоровья сотрудника после прохождения обследования. Данные о врачах, которые составляют заключение по результатам обследования.

В данной базе данных реализованы механизмы защиты от некорректного ввода данных с помощью триггеров. В базе данных существуют триггеры на проверку фамилий, имен и отчеств, как сотрудников компании, так и врачей, на проверку даты проведения медосмотра, на проверку корректного ввода номера телефона сотрудника и его паспортных данных.

Также в базе данных реализовано много хранимых процедур, которые позволяют выводить нужную информацию для пользователя системы, удалять, обновлять и вставлять новые данные о сотрудниках, их паспортных данных и медицинских осмотрах. Помимо прочего, существует процедура, подсчитывающая количество сотрудников, по итогам обследования которых они могут быть допущены к работе (рейсу).

Созданную систему можно улучшить, расширяя уже имеющийся и добавляя новый функционал. Так, например, можно добавить возможность автоматического удаления данных о медицинских осмотрах, которые проводились давно и имеют уже неактуальную информацию.

Модернизация системы может произойти с добавлением новых хранимых процедур и триггеров, а также совершенствованием уже имеющихся.

1. Построение системы защиты данных
2. Создание регистрационных имен, имен пользователей и ролей

Защита информации – комплекс мероприятий, направленных на обеспечение важнейших аспектов информационной безопасности (целостности, доступности и конфиденциальности информации и ресурсов, используемых для ввода, хранения, обработки и передачи данных).

Система называется безопасной, если она, используя соответствующие аппаратные и программные средства, управляет доступом к информации так, что только должным образом авторизованные лица или же действующие от их имени процессы получают право читать, писать, создавать и удалять информацию.

Создать регистрационные имена можно двумя способами: с помощью среды Management Studio или с использованием инструкции языка Transact-SQL.

С помощью выполнения инструкций языка Transact-SQL создаются регистрационные имена employee01, employee02, employee03, employee04, employee05, employee06, employee07, employee08 и employee09 с соответствующими паролями VerySTRPass1, VerySTRPass2, VerySTRPass3, VerySTRPass4, VerySTRPass5, VerySTRPass6, VerySTRPass7, VerySTRPass8 и VerySTRPass9 в базе данных автоматизированной информационной системы «Медицинское обследование в удаленном режиме».

Создание регистрационных имен с соответствующими им паролями изображено в приложении на рисунке 51.

Проверка отображения имен в Обозревателе объектов (системном каталоге) показана в приложении на рисунке 52.

Для регистрационных имен (логинов) employee01, employee02, employee03, employee04, employee05, employee06, employee07, employee08 и employee09 создаются соответствующие имена пользователей emp\_muhanov, emp\_romanko, emp\_krutov, emp\_petrov, emp\_sidorov, emp\_simonenko, emp\_kozlov, emp\_smirnova и emp\_nikolaev базы данных MedExamSys путем выполнения инструкций языка Transact-SQL. Создание имен пользователей показано в приложении на рисунке 53.

Проверка отображения пользователей в Обозревателе объектов (системном каталоге) показана в приложении на рисунке 54.

Создадим новую определяемую пользователем роль directors базы данных MedExamSys и добавим в нее пользователя emp\_muhanov с помощью инструкций языка Transact-SQL. Создание роли и добавление пользователя показано в приложении на рисунке 55.

Создадим новую определяемую пользователем роль admins базы данных MedExamSys и добавим в нее пользователей emp\_romanko и emp\_krutov с помощью инструкций языка Transact-SQL. Создание роли и добавление пользователей показано в приложении на рисунке 56.

Создадим новую определяемую пользователем роль doctors базы данных MedExamSys и добавим в нее пользователей emp\_petrov, emp\_sidorov, emp\_simonenko, emp\_kozlov, emp\_smirnova и emp\_nikolaev с помощью инструкций языка Transact-SQL. Создание роли и добавление пользователей показано изображены в приложении на рисунке 57.

Проверка отображения ролей в Обозревателе объектов (системном каталоге) показана в приложении на рисунке 58.

1. Работа и реализация разрешений в базе данных

Для реализации разрешений используются инструкции языка Transact-SQL.

Используя инструкцию GRANT, предоставим роли admins разрешение на создание таблиц и хранимых процедур, а также на выполнение хранимых процедур Insert\_med\_check, Delete\_emp, Update\_pass\_id и Update\_data\_pass в базе данных автоматизированной системы «Медицинское обследование в удаленном режиме».

Предоставление всех выше описанных разрешений роли admins показано на рисунке 2.1.

Проверка отображения разрешений в свойствах базы данных изображено в приложении на рисунке 59, а в свойствах хранимых процедур в приложении на рисунках 60-63.

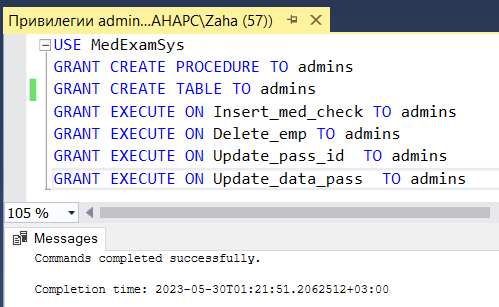


Рисунок 2.1 - Предоставление разрешений роли admins

Используя инструкцию GRANT, предоставим роли directors разрешение на добавление и выборку данных из таблицы Passport и Employee, а также на выполнение хранимых процедур Select\_concl\_for\_date, Select\_res\_for\_emp, Select\_emp\_ready, Update\_data\_pass, Delete\_emp, Update\_pass\_id и Calc\_emp\_ready в базе данных автоматизированной системы «Медицинское обследование в удаленном режиме».

Предоставление всех выше описанных разрешений роли directors показано на рисунке 2.2.

Проверка отображения разрешений в свойствах таблиц изображено в приложении на рисунках 64-65, а в свойствах хранимых процедур в приложении на рисунках 66-72.

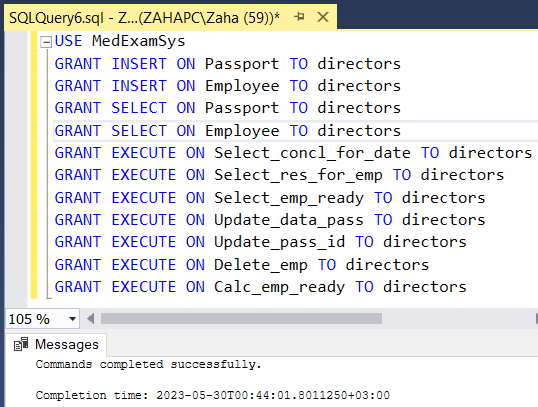


Рисунок 2.2 - Предоставление разрешений роли directors

Используя инструкцию GRANT, предоставим роли doctors разрешение на добавление данных в таблицу Protocol, а также на выполнение хранимой процедуры Select\_res\_for\_emp в базе данных автоматизированной системы «Медицинское обследование в удаленном режиме».

Предоставление всех выше описанных разрешений роли doctors показано на рисунке 2.3.

Проверка отображения разрешений в свойствах таблиц изображено в приложении на рисунке 73, а в свойствах хранимых процедур в приложении на рисунке 74.

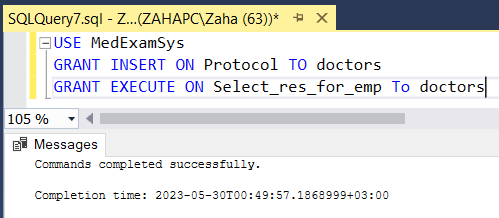


Рисунок 2.3 - Предоставление разрешений роли doctors

При выдаче различных привилегий ролям, важно отметить, что в случае, если необходимо запретить пользователям выполнение некоторых операций, необходимо воспользоваться оператором DENY. Так, например, роли doctors запрещается операция SELECT на выборку данных из таблиц Employee и Passport, что изображено на рисунке 2.4.

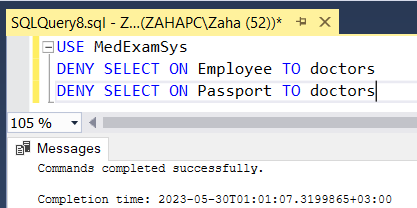


Рисунок 2.4 - Запрет операции SELECT для таблиц Employee и Passport роли doctors

В случае, если необходимо отобрать ранее выданное разрешение, либо запрет, необходимо воспользоваться инструкцией REVOKE. В примере ниже на рисунке 2.5 показан пример её использования.

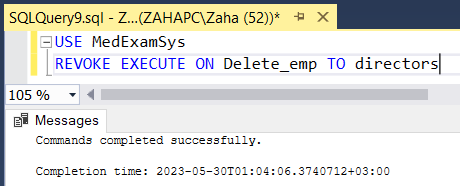


Рисунок 2.5 - Отбор разрешения на выполнение процедуры Delete\_emp у роли directors

С помощью системной процедуры sp\_helpuser получим информацию о пользователях emp\_muhanov, emp\_romanko, emp\_krutov, emp\_petrov, emp\_sidorov, emp\_simonenko, emp\_kozlov, emp\_smirnova и emp\_nikolaev касательно базы данных MedExamSys, что показано на рисунке 2.6.

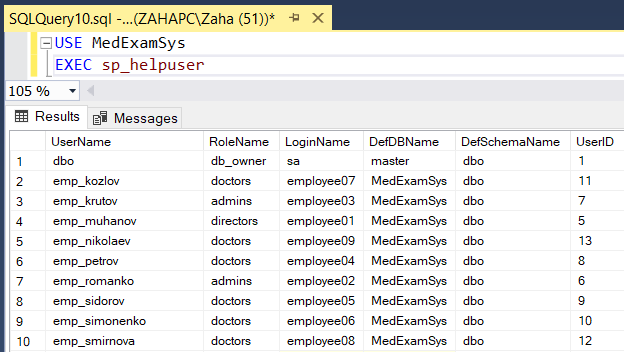


Рисунок 2.6 - Получение информации о пользователях базы данных

1. Применение и реализация шифрования в базе данных

Шифрование представляет собой способ скрытия данных с помощью ключа или пароля. Это делает данные бесполезными без соответствующего ключа или пароля для дешифрования. Шифрование повышает защиту за счет ограничения потери данных даже при обходе системы управления доступом.

Шифрование можно выполнять различными способами. В данной работе реализованы способы шифрования базы данных на уровне столбца и с использованием прозрачного шифрования.

Перед началом шифрования необходимо создать ключ Database Master Key (DMK) базы данных автоматизированной информационной системы «Медицинское обследование в удаленном режиме», который шифруется паролем DMKbdMedExamSys2023. Создание зашифрованного ключа DMK в базе данных показано на рисунке 2.7.

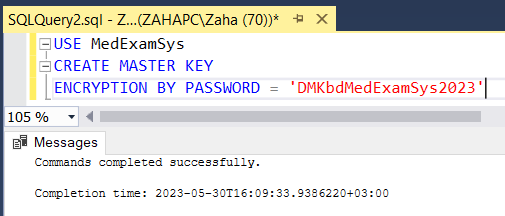


Рисунок 2.7 - Создание зашифрованного ключа DMK

С помощью DMK шифруются симметричные и асимметричные ключи, а также и сертификаты. Каждая база данных может располагать лишь одним DMK.

После создания DMK в базе данных MedExamSys создается асимметричный ключ ASMKEY с алгоритмом шифрования RSA\_2048, зашифрованный паролем ASYMKEYMedExamSys2023.

Асимметричный ключ является защищаемой сущностью на уровне базы данных. По умолчанию эта сущность содержит как открытый, так и закрытый ключ. Закрытый ключ защищается с помощью главного ключа базы данных. Для защиты закрытого ключа необходим пароль, если не был создан главный ключ базы данных. Если главный ключ базы данных существует, пароль необязателен. Создание асимметричного ключа ASMKEY показано на рисунке 2.8.

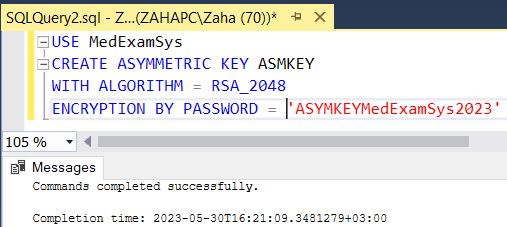


Рисунок 2.8 - Создание ключа ASMKEY

С помощью системного представления sys.asymmetric\_keys можно убедиться, что асимметричный ключ создан. Проверка того, что асимметричный ключ создан показана на рисунке 2.9.

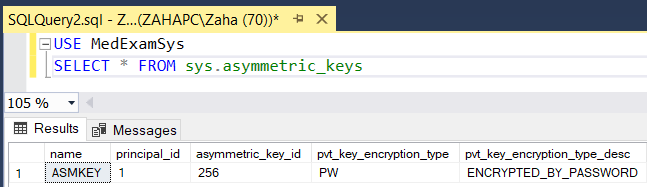


Рисунок 2.9 - Проверка создания асимметричного ключа

Далее создается симметричный ключ SYMKEY, который шифруется с помощью раннее созданного асимметричного ключа ASMKEY, с алгоритмом шифрования AES256.

Создание симметричного ключа SYMKEY показано на рисунке 2.10.



Рисунок 2.10 - Создание ключа SYMKEY

С помощью представления sys.symmetric\_keys можно убедиться, что симметричный ключ создан. Проверка того, что симметричный ключ создан показано на рисунке 2.11.

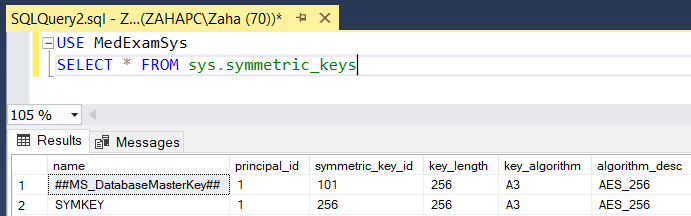


Рисунок 2.11 – Проверка создания симметричного ключа

После создания симметричный ключ должен быть зашифрован с помощью одного из следующих средств: сертификат, пароль, симметричный ключ, асимметричный ключ или PROVIDER. Один симметричный ключ может быть зашифрован с использованием нескольких сертификатов, симметричных ключей и асимметричных ключей одновременно.

Чтобы зашифровать данные таблицы Employee необходимо выполнить ряд запросов. Для начала необходимо открыть симметричный ключ SYMKEY для шифрования данных.

Открытие ключа SYMKEY показано на рисунке 2.12.

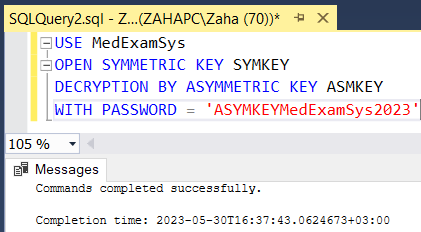


Рисунок 2.12 - Открытие ключа SYMKEY

Проверка того, что ключ открыт с помощью системного представления sys.openkeys показана на рисунке 2.13.

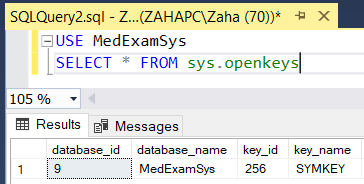


Рисунок 2.13 - Проверка открытия ключа

Далее создается столбец emp\_enc\_phone в таблице Employee, в котором будут храниться зашифрованные данные из столбца emp\_phone. С помощью функции ENCRYPTBYKEY производится шифрование столбца Номер\_телефона\_сотрудника при использовании симметричного ключа SYMKEY.

Создание столбца emp\_enc\_phone и шифрование столбца emp\_phone показаны на рисунке 2.14. Проверка зашифрованных данных из таблицы Employee показана на рисунке 2.15.

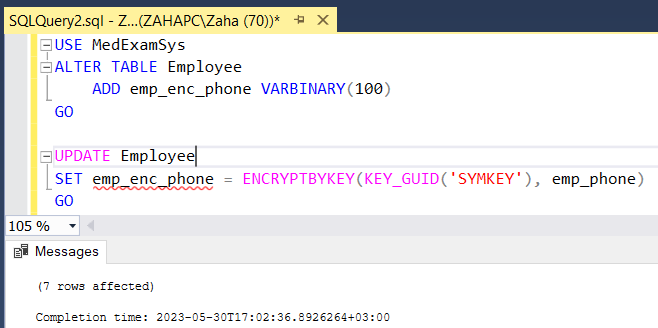


Рисунок 2.14 - Создание столбца emp\_enc\_phone и шифрование столбца emp\_phone

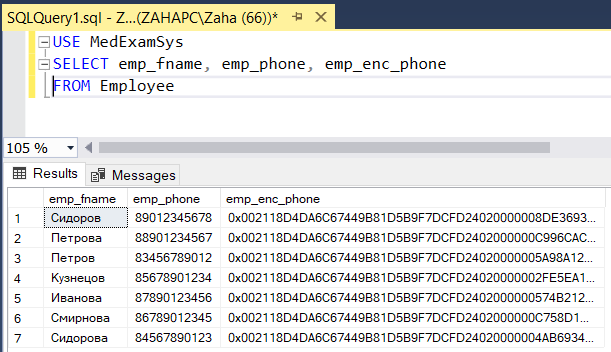


Рисунок 2.15 - Проврека зашифрованных данных

Проверка с помощью функции DECRYPTBYKEY того, что данные в столбце emp\_enc\_phone действительно являются зашифрованными данными столбца emp\_phone изображена на рисунке 2.16.

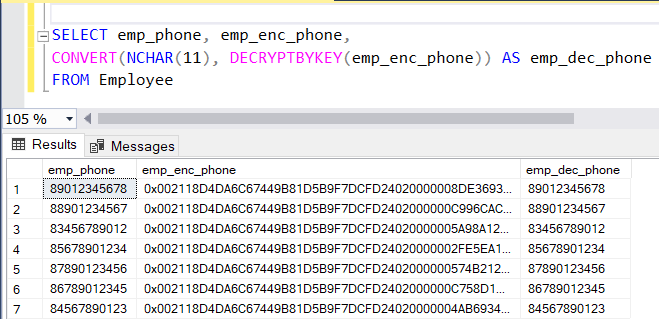


Рисунок 2.16 - Дешифрование столбца emp\_enc\_phone

После шифрования столбца таблицы с помощью процедур зашифруем всю базу данных с использованием прозрачного шифрования.

Функция прозрачного шифрования данных выполняет шифрование и дешифрование ввода-вывода в реальном времени для файлов данных и журналов. Шифрование использует ключ шифрования базы данных (DEK). DEK является симметричным ключом. Он защищен сертификатом, который хранится в базе данных сервера master, или асимметричным ключом, который защищает модуль расширенного управления ключами.

При включении функции прозрачного шифрования данных необходимо немедленно создать резервную копию сертификата и закрытого ключа, связанного с этим сертификатом. В случае если сертификат окажется недоступен или понадобится восстановить базу данных на другом сервере либо присоединить ее к другому серверу, необходимо иметь копии сертификата и закрытого ключа. В противном случае будет невозможно открыть базу данных.

Создание главного ключа шифрования, его резервной копии и размещение ее в надежном месте, а также просмотр ключа с помощью системного представления sys.key\_encryptions показано на рисунке 2.17.

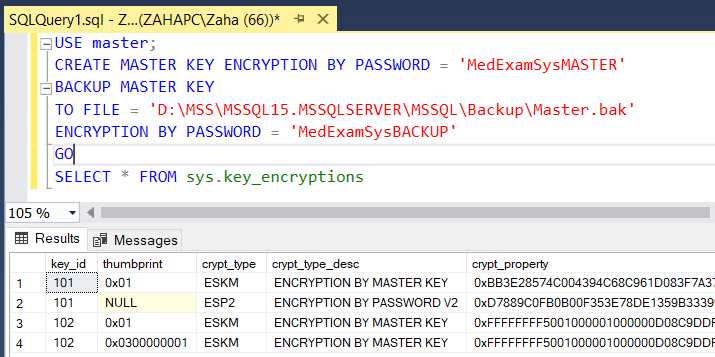


Рисунок 2.17 - Создание главного ключа шифрования

Далее необходимо создать сертификат MedExamSysCERT, при этом также создается его резервная копия и помещается в надежное место. Создание сертификата, его резервной копии, помещение ее в надежное место, а также просмотр сертификата с помощью системного представления sys.certificates показано на рисунке 2.18.

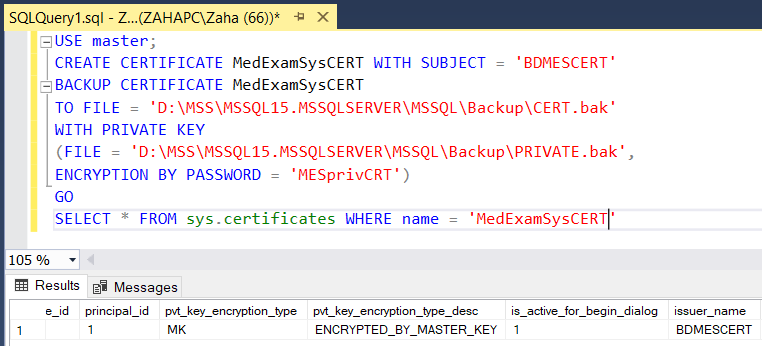


Рисунок 2.18 - Создание сертификата MedExamSysCERT с резервной копией

Проверка наличия файлов резервной копии главного ключа шифрования и сертификата с закрытым ключом в папке диска, которая была выбрана как надежное место в момент создания резервной копии изображена на рисунке 2.19.

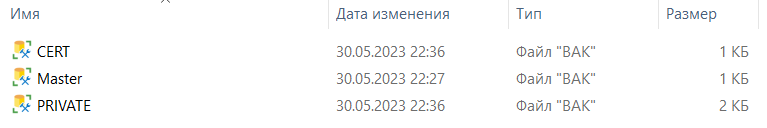


Рисунок 2.19 - Проверка наличия резервных копий

Сертификат - это цифровое предложение с подписью, которое связывает открытый ключ с определенным лицом, устройством или службой, которая владеет соответствующим открытым ключом.

Сертификаты практически не отличаются от асимметричных ключей, так как и те, и другие используют алгоритм RSA. Единственное различие заключается в том, что асимметричные ключи создаются вне сервера.

Шифрование базы данных будет осуществляется с помощью алгоритма AES с длиной ключа 128 бит. Данный алгоритм шифрования является симметричным алгоритмом блочного шифрования и в настоящее время признан надежным и все современные системы безопасности должны использовать этот алгоритм.

Следующим этапом в шифровании базы данных является создание ключа шифрования в базе данных MedExamSys с использованием сертификата MedExamSysCERT. После создания ключа необходимо включить шифрование с помощью команды SET ENCRYPTION ON.

С помощью системного представления sys.dm\_database\_encryption\_keys можно проверить, что ключ шифрования в базе данных автоматизированной информационной системы «Медицинское обследование в удаленном режиме» действительно создан.

Создание ключа шифрования базы данных MedExamSys , а также включение шифрования показаны на рисунке 2.20.

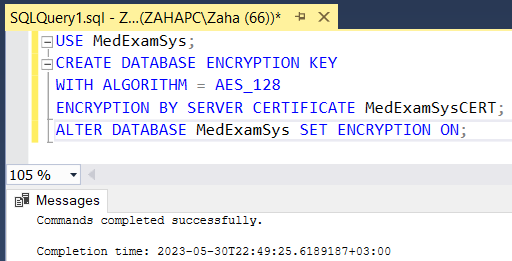


Рисунок 2.20 - Создание ключа шифрования базы данных MedExamSys

Проверка создания ключа изображена на рисунке 2.21.

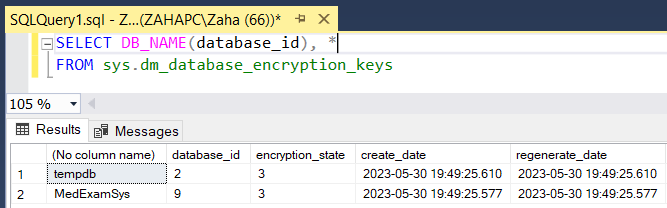


Рисунок 2.21 - Проверка создания ключа базы данных MedExamSys

Далее начинается процесс первоначального шифрования базы данных MedExamSys. Этот процесс выполняется «в фоне» в отдельном потоке. Отследить процесс выполнения этой операции можно по столбцу percent\_complete системного представления sys.dm\_database\_encryption\_keys.

Отслеживание процесса шифрования показано на рисунке 2.22.

В столбце encryption\_state содержится информация о текущем состоянии базы данных. В контексте Transparent Data Encryption (TDE) БД может находиться в одном из следующих состояний:

0 - DEK не создан;

1 - DEK создан, но база данных не зашифрована;

2 - Выполняется первоначальное шифрование;

3 - База данных зашифрована;

4 - Идет смена ключа;

5 - Идет расшифровка.

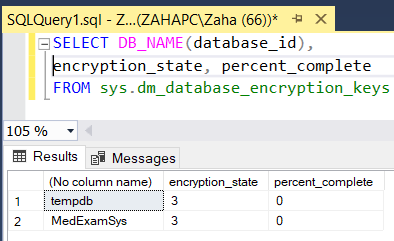


Рисунок 2.22 - Отслеживание процесса шифрования

Таким образом, согласно столбцу encryption\_state шифрование базы данных завершено.

1. Выводы по второй главе

Во второй главе курсовой работы была рассмотрена разработка системы защиты базы для хранения и обработки данных, полученных при проведении медицинского обследования в удаленном режиме сотрудников транспортной компании.

Для обеспечения безопасности базы данных были использованы регистрационные имена, пользователи и роли. Пользователи были разделены на группы в зависимости от выполняемых задач. Роли были назначены каждой группе, чтобы пользователи могли иметь привилегии в базе данных. Важным аспектом обеспечения безопасности базы данных является назначение привилегий для пользователей в зависимости от их роли.

Также для обеспечения безопасности базы данных было применено прозрачное шифрование и шифрование столбцов таблиц. Прозрачное шифрование данных позволяет защитить данные в базе данных, не нарушая ее работоспособность. Шифрование столбцов позволяет защитить конфиденциальные данные от несанкционированного доступа.

Таким образом, в данной части работы были рассмотрены различные аспекты обеспечения безопасности базы данных для хранения и обработки данных, полученных при проведении медицинского обследования в удаленном режиме сотрудников транспортной компании. Результатом является разработка базы данных с высоким уровнем безопасности и защиты данных.

Заключение

Исходя из предоставленной выше информации, можно сделать вывод, что разрабатываемая база данных является полезной и необходимой платформой. А полностью описанный способ ее реализации позволит с легкостью воспроизвести ее для общего пользования на рынок. Данная база данных использует оптимально ресурсы и имеет возможность для дальнейшей модернизации.

В данной курсовой работе проведена разработка и защита базы данных для хранения и обработки данных, полученных при проведении медицинского обследования в удаленном режиме сотрудников транспортной компании. В ней есть все необходимые инструменты для быстрого поиска и изменение данных. Для достижения поставленной цели были рассмотрены и решены следующие основные задачи по:

1. анализу предметной области и разработке требований к базе данных информационной системы «Медицинское обследование в удаленном режиме»;
2. проектированию подмоделей логического и физического уровней представления данных;
3. реализации полученной модели данных в СУБД SQL Server 2019 Developer;
4. построению системы защиты базы данных.

Список литературы

1. Федеральный закон "О внесении изменений в статью 46 Федерального закона "Об основах охраны здоровья граждан в Российской Федерации" и статью 23 Федерального закона "О безопасности дорожного движения" от 29.12.2022 N 629-ФЗ.
2. Голицына, О. Л. Базы данных. Учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - М.: Дрофа, 2019. - 400 c.
3. Бен-Ган, Ицик Microsoft SQL Server 2012. Основы T-SQL / Ицик Бен-Ган. — М.: Эксмо, 2021. — 759 c.
4. Фуфаев, Э.В. Базы данных / Э.В. Фуфаев, Д.Э. Фуфаев. - М.: Академия, 2019. – 320 c.;
5. Советов Б.Я. Базы данных: учебник для прикладного бакалавриата / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. – М.: Издательство «Юрайт», 2020. – 463 с.
6. Microsoft SQL Server 2019. Руководство для начинающих / Петкович Д. — NY.: McGraw Hill, 2020. — 864 с.
7. SQL: быстрое погружение / Шилдс Уолтер — СПб.: Питер, 2022. — 224 с.
8. Programming For Data Science: 2 Books in 1: Cyber Security, SQL Programming / Ben Chan — Washington.: Ben Chan, 2021. — 607 с.
9. Сарка, Деян Microsoft SQL Server 2012. Реализация хранилищ данных. Учебный курс Microsoft (+ CD-ROM) / Деян Сарка , Матия Лах , Грега Йеркич. — Москва: Высшая школа, 2020. — 816 c.

**Приложение А**

Рисунки к главе 1:



Рисунок 1 - Контекстная диаграмма

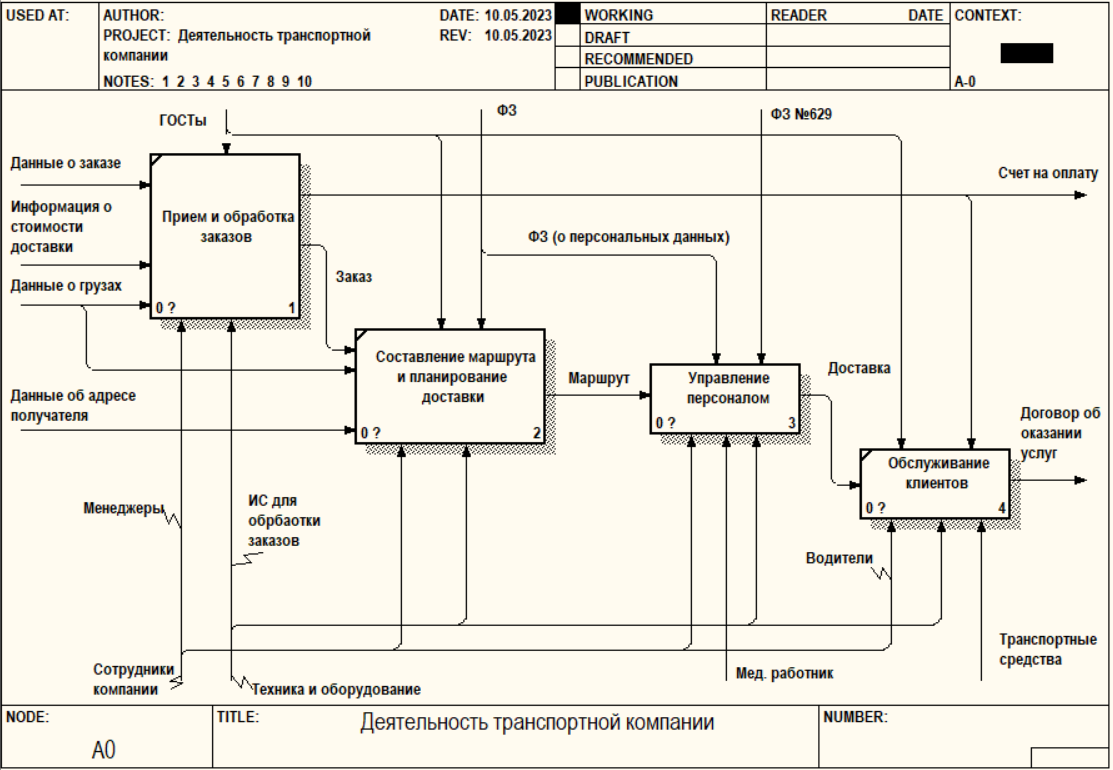


Рисунок 2 - Диаграмма декомпозиции первого уровня

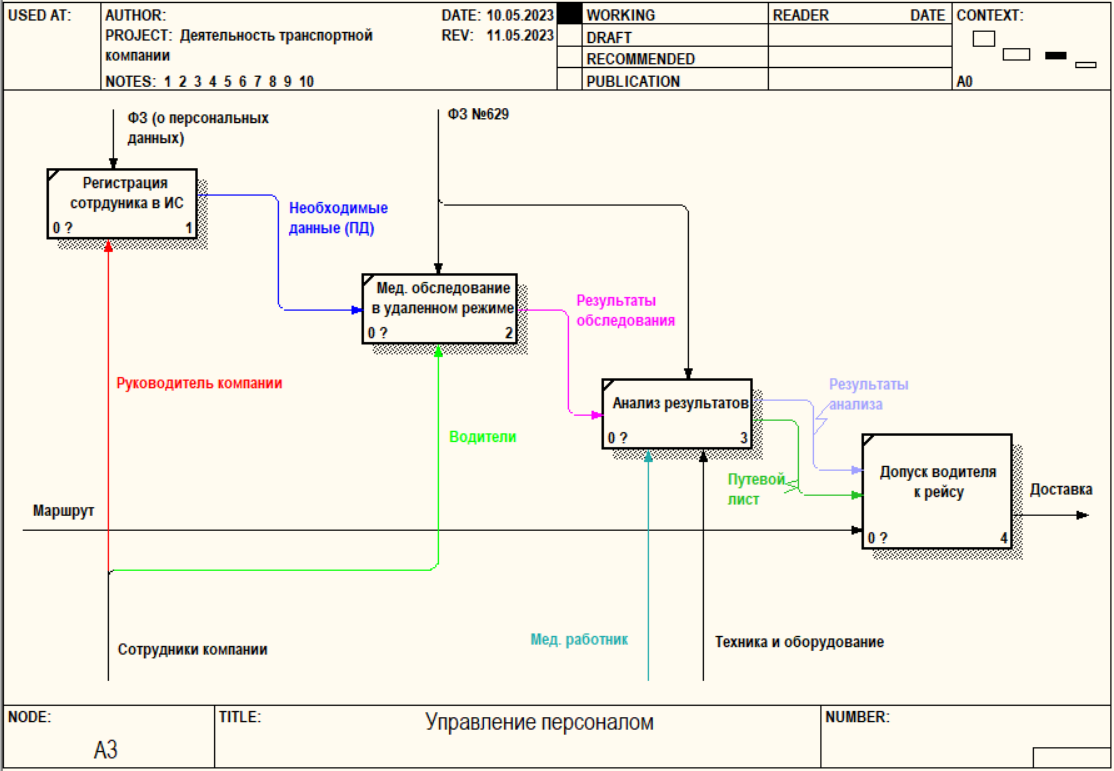


Рисунок 3 - Диаграмма декомпозиции второго уровня

Т а б л и ц а 1 – Свойства атрибутов отношений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Отношение | Атрибуты отношения | Тип атрибута | Свойства атрибута | Комментарий |
| Employee | emp\_id | INT | NOT NULL | Первичный ключ; серия и номер паспорта |
| emp\_fname | CHAR(30) | NOT NULL |  |
| emp\_name | CHAR(30) | NOT NULL |  |
| emp\_surname | CHAR(30) | NULL |  |
| emp\_age | CHAR(3) | NOT NULL | Поле типа CHAR, но содержит цифры, из-за того, что с ним не нужно выполнять вычислений. |
| emp\_gender | CHAR(1) | NOT NULL | Поле со списком, который состоит: «М», «Ж». |
| emp\_phone | NCHAR(11) | NOT NULL | Поле типа NCHAR, но содержит цифры, из-за того, что с ним не нужно выполнять вычислений. |
| emp\_email | NCHAR(20) | NULL |  |
| Passport | pass\_id | NCHAR(11) | NOT NULL | Первичный ключ, серия и номер паспорта. |
| pass\_date | DATE | NOT NULL |  |
| pass\_issued | VARCHAR | NOT NULL |  |
| pass\_code | NCHAR(7) | NOT NULL |  |
| MedCheckup | check\_id | INT | NOT NULL | Первичный ключ; счетчик. |
| check\_date | DATATIME | NOT NULL | Значение по умолчанию – сегодняшняя дата. |
| Doctor | doc\_id | INT | NOT NULL | Первичный ключ; счетчик. |
| doc\_fname | CHAR(30) | NOT NULL |  |
| doc\_name | CHAR(30) | NOT NULL |  |
| doc\_surname | CHAR(30) | NULL |  |
| doc\_spec | CHAR(20) | NOT NULL |  |
| doc\_email | NCHAR(20) | NULL |  |
| Results | res\_temp | FLOAT | NOT NULL |  |
| res\_sys\_pres | CHAR(3) | NOT NULL | Поле типа CHAR, но содержит цифры, из-за того, что с ним не нужно выполнять вычислений. |
| res\_dias\_pres | CHAR(3) | NOT NULL | Поле типа CHAR, но содержит цифры, из-за того, что с ним не нужно выполнять вычислений. |
| res\_pulse | CHAR(3) | NOT NULL | Поле типа CHAR, но содержит цифры, из-за того, что с ним не нужно выполнять вычислений. |
| res\_alco | FLOAT | NOT NULL |  |
| res\_compl | NCHAR(100) | NULL |  |
| Protocol | prot\_num | CHAR(5) | NOT NULL | Первичный ключ, уникальный номер протокола. |
| prot\_concl | NCHAR(100) | NOT NULL |  |

Т а б л и ц а 2 – Ключи и индексы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Отношение | Атрибут | Тип | Индекс | Комментарий |
| Employee | emp\_id | FK | Index | Служит для связи с отношением Passport |
| emp\_fname |  | Index |  |
| emp\_name |  | Index |  |
| Passport | pass\_id | PK | Index Unique |  |
| MedCheckup | check\_id | PK | Index Unique |  |
| emp\_id | FK | Index | Служит для связи с отношением Employee |
| doc\_id | FK | Index | Служит для связи с отношением Doctor |
| Doctor | doc\_id | PK | Index Unique |  |
| doc \_fname |  | Index |  |
| doc \_name |  | Index |  |
| Results | res\_id | FK | Index | Служит для связи с отношением MedCheckup |
| prot\_num | FK | Index | Служит для связи с отношением Protocol. |
| Protocol | prot\_num | PK | Index Unique |  |

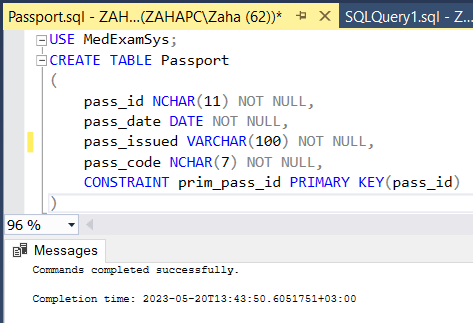


Рисунок 4 - Создание таблицы Passport

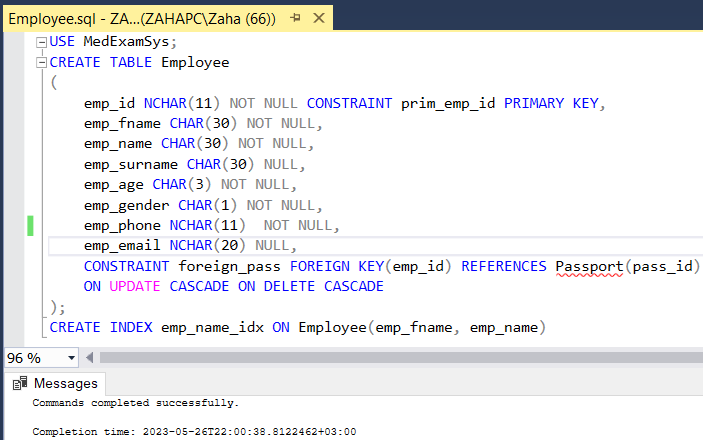


Рисунок 5 - Создание таблицы Employee.



Рисунок 6 - Создание таблицы Doctor

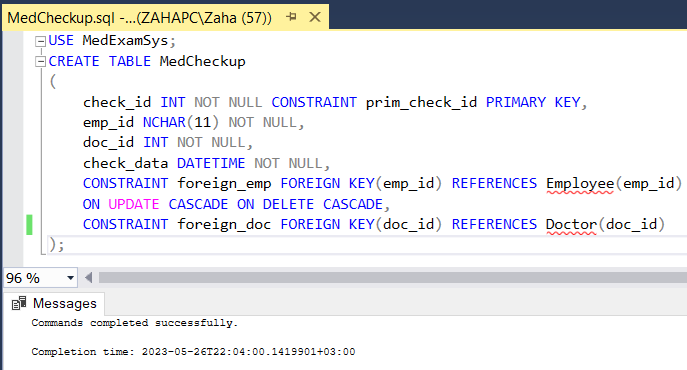


Рисунок 7 - Создание таблицы MedCheckup

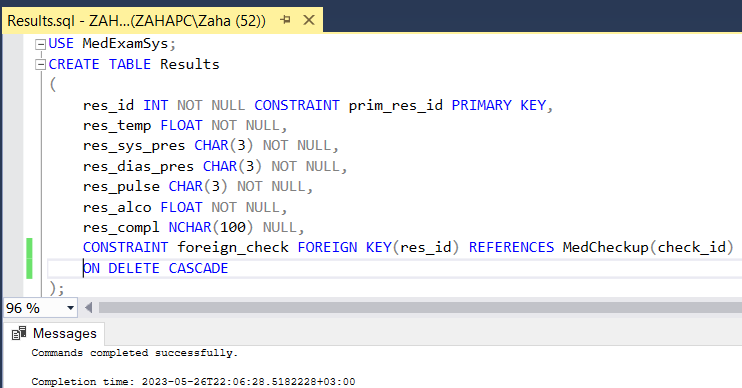


Рисунок 8 - Создание таблицы Results

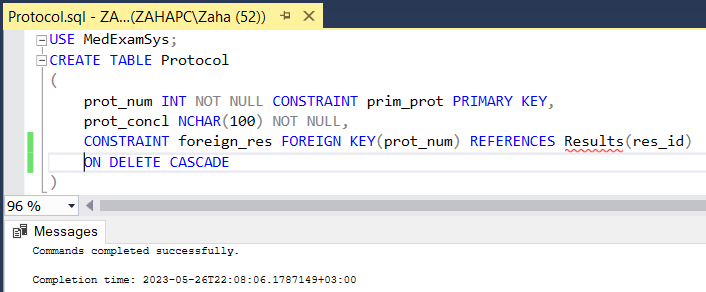


Рисунок 9 - Создание таблицы Protocol

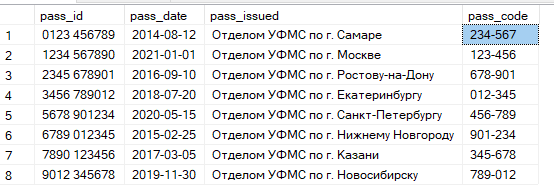


Рисунок 10 - Заполненная таблица Passport.



Рисунок 11 - Заполненная таблица Employee

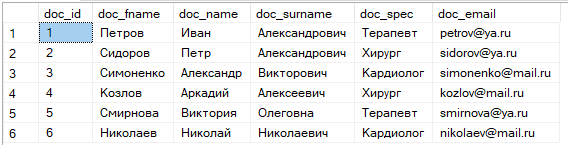


Рисунок 12 - Заполненная таблица Doctor

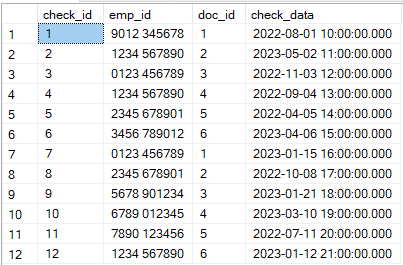


Рисунок 13 - Заполненная таблица MedCheckup

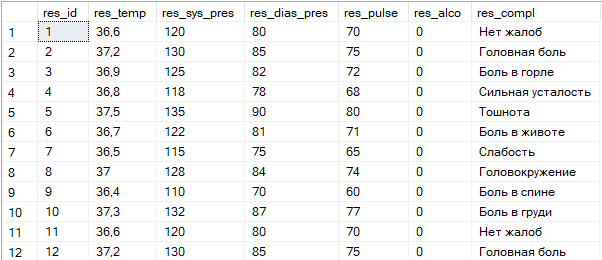


Рисунок 14 - Заполненная таблица Results

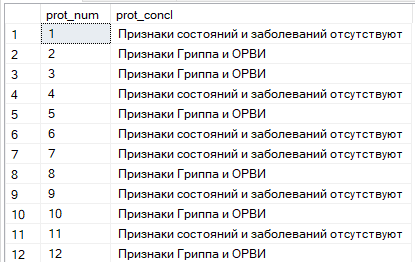


Рисунок 15 - Заполненная таблица Protocol

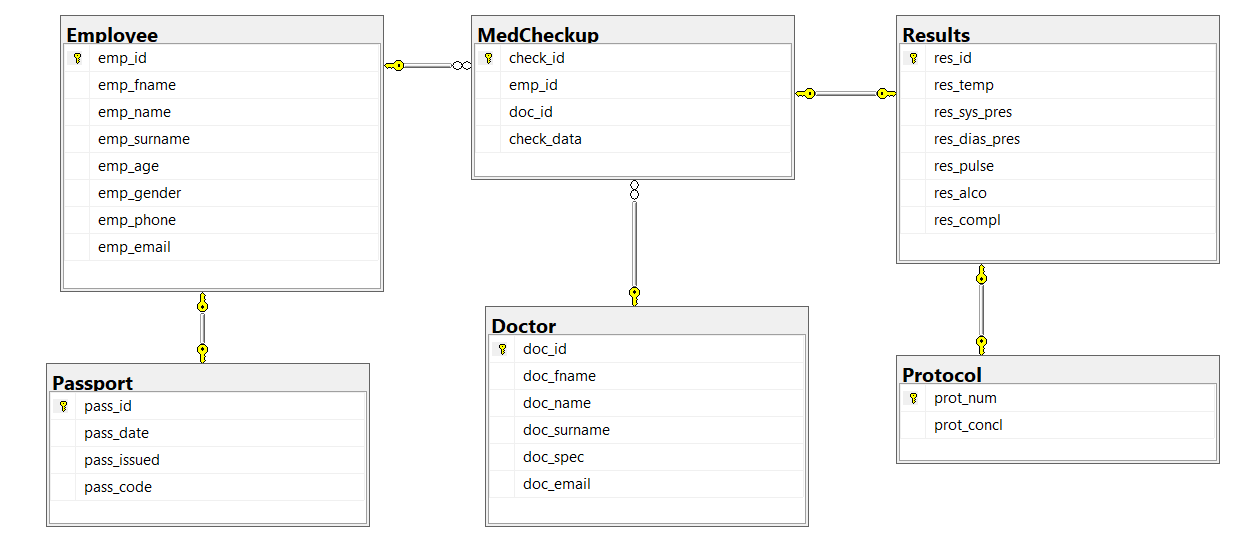


Рисунок 16 - Диаграмма базы данных MedExamSys

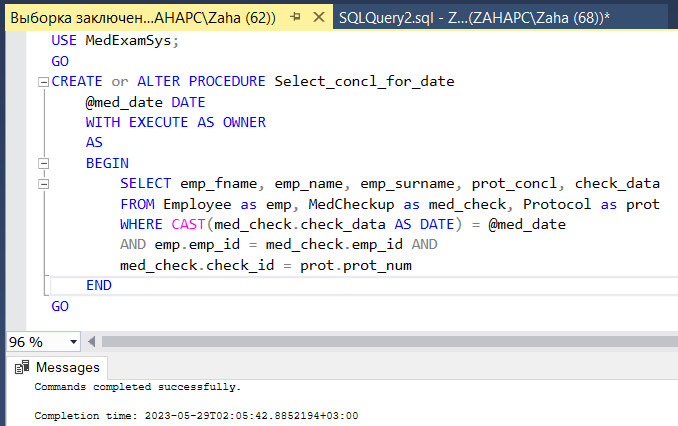


Рисунок 17 - Создание процедуры Select\_concl\_for\_date

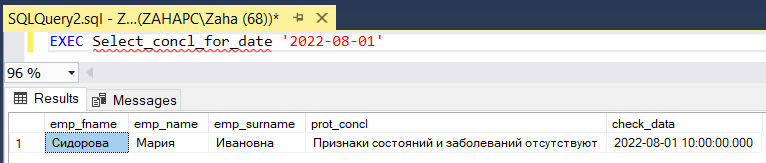


Рисунок 18 - Пример работы процедуры Select\_concl\_for\_date

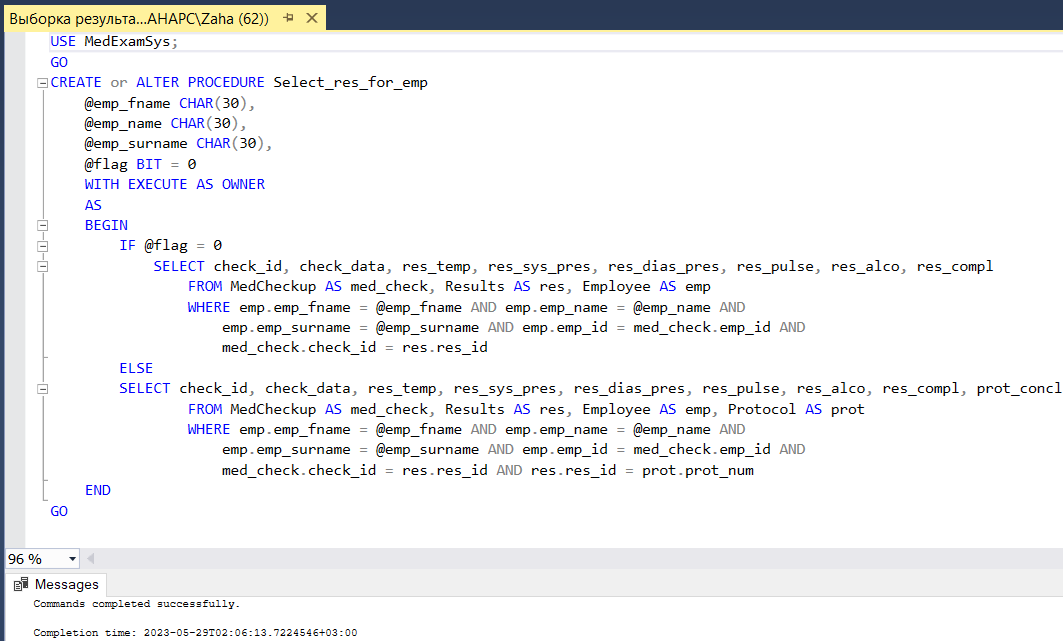


Рисунок 19 - Создание процедуры Select\_res\_for\_emp

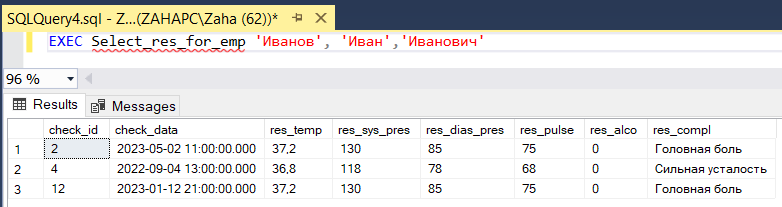


Рисунок 20 - Пример работы процедуры Select\_res\_for\_emp

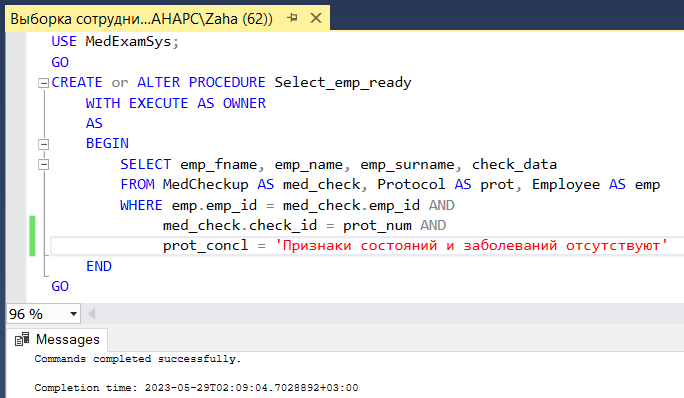


Рисунок 21 - Создание процедуры Select\_emp\_ready



Рисунок 22 - Пример работы процедуры Select\_emp\_ready

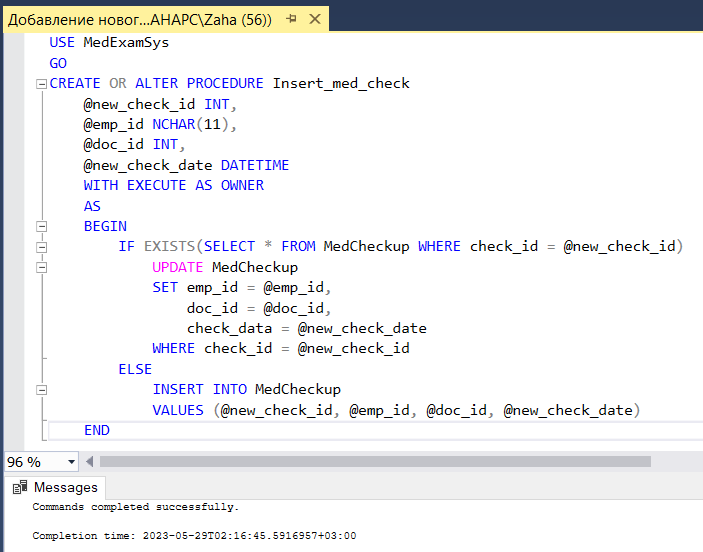


Рисунок 23 - Создание процедуры Insert\_med\_check

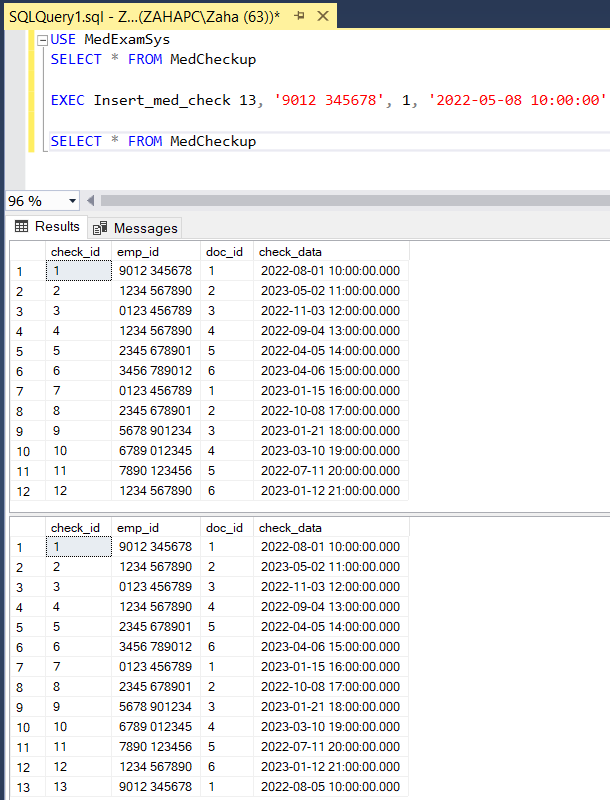


Рисунок 24 - Пример работы процедуры Insert\_med\_check

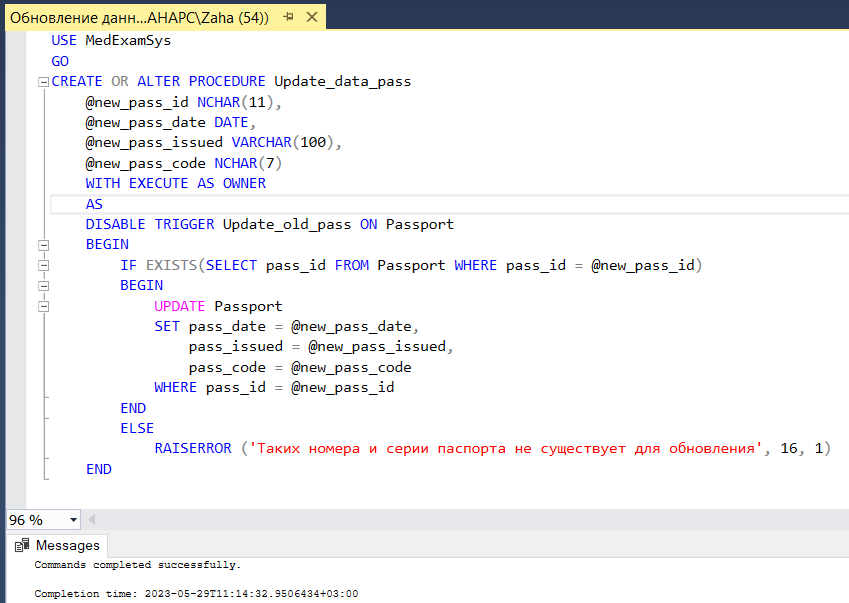


Рисунок 25 - Создание процедуры Update\_data\_pass

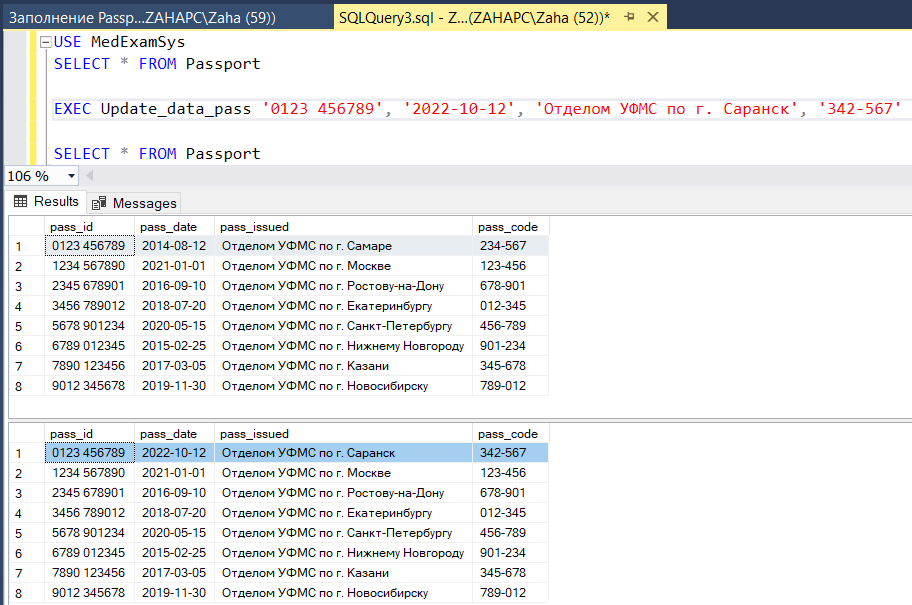


Рисунок 26 - Первый пример работы процедуры Update\_data\_pass

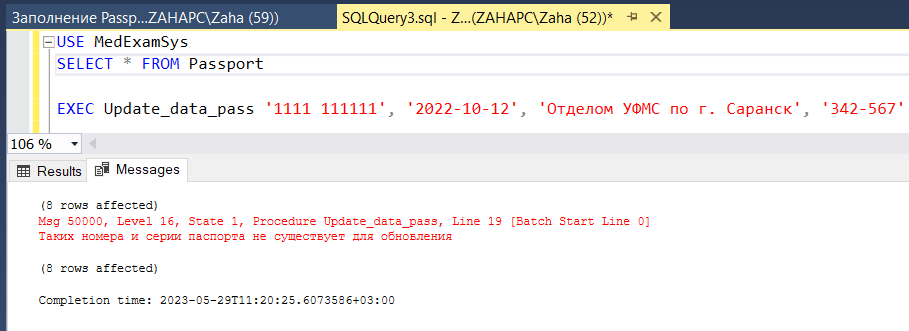


Рисунок 27 - Второй пример работы процедуры Update\_data\_pass

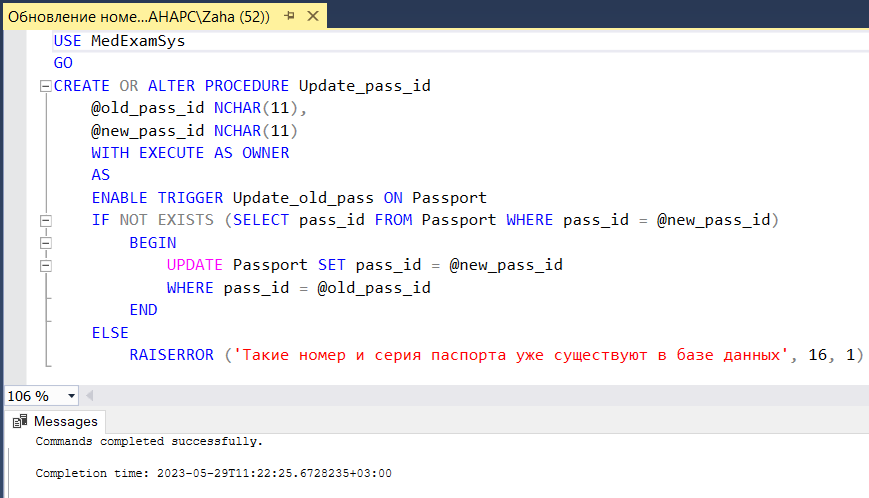


Рисунок 28 - Создание процедуры Update\_pass\_id

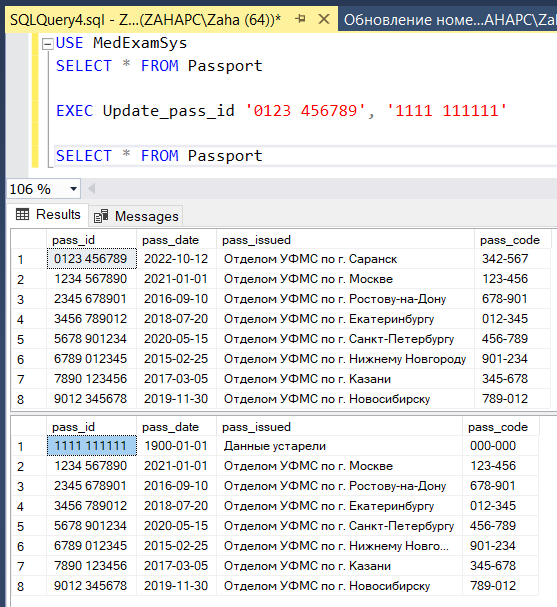


Рисунок 29 - Первый пример работы процедуры Update\_pass\_id

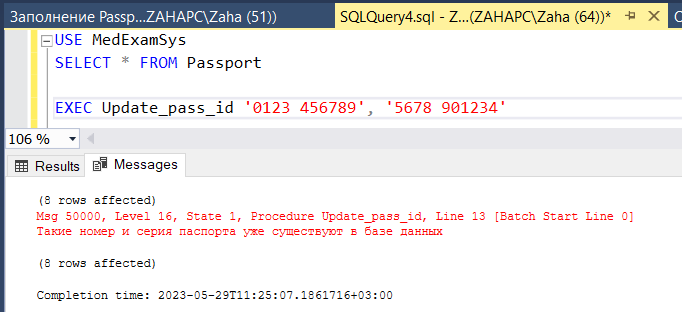


Рисунок 30 - Второй пример работы процедуры Update\_pass\_id

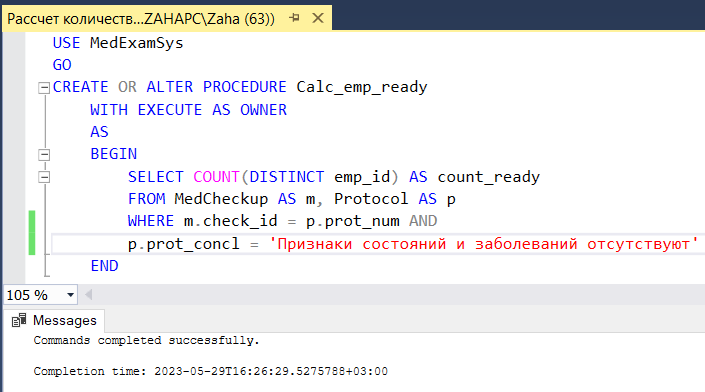


Рисунок 31 - Создание процедуры Calc\_emp\_ready

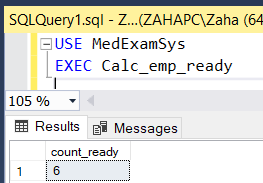


Рисунок 32 - Пример работы процедуры Calc\_emp\_ready

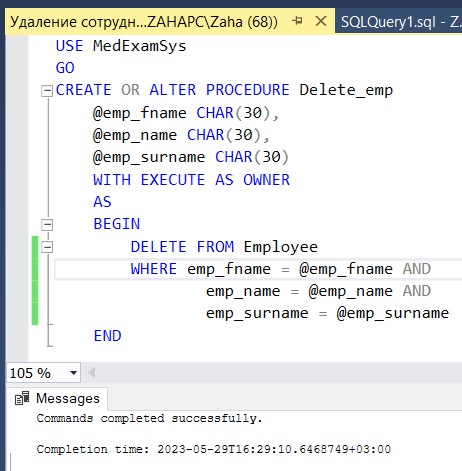


Рисунок 33 - Создание процедуры Delete\_emp

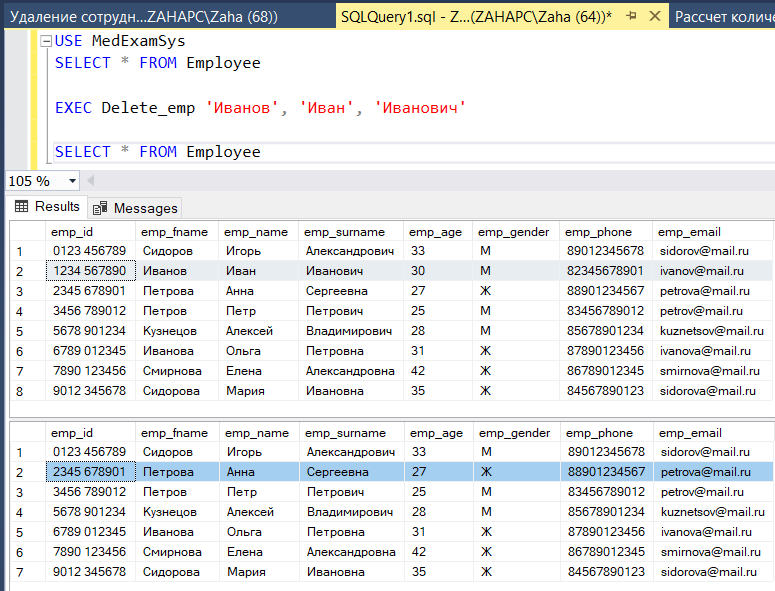


Рисунок 34 - Пример работы процедуры Delete\_emp

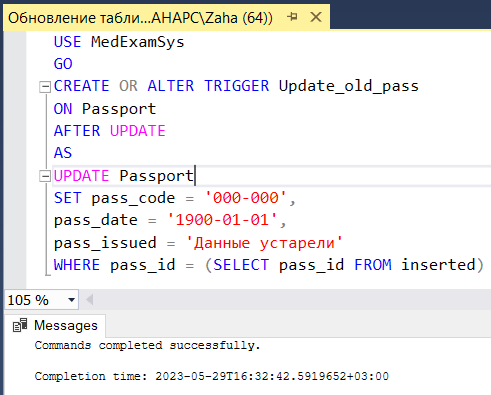


Рисунок 35 - Создание триггера Update\_old\_pass

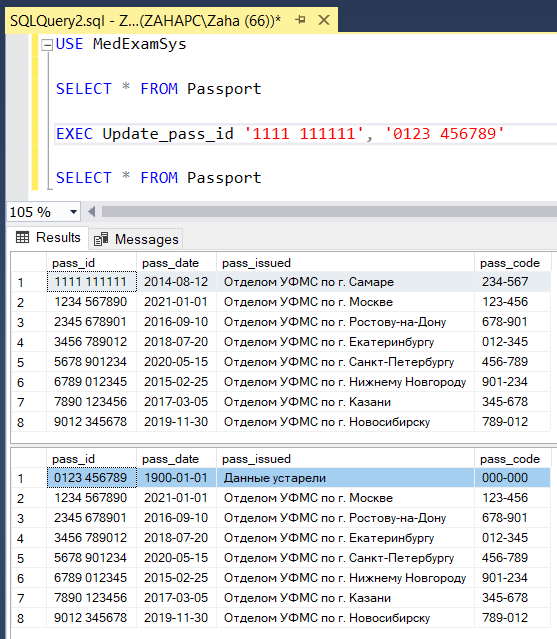


Рисунок 36 - Пример выполнения триггера Update\_old\_pass



Рисунок 37 - Создание триггера Insert\_prot\_concl

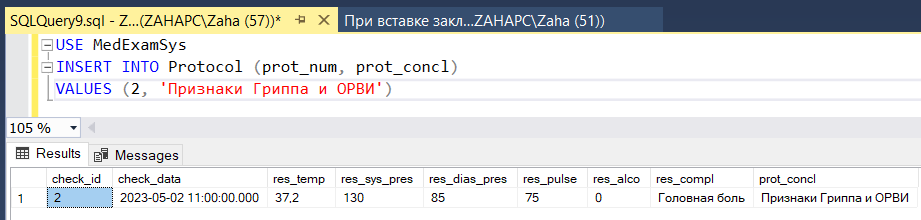


Рисунок 38 - Пример выплнения триггера Insert\_prot\_concl

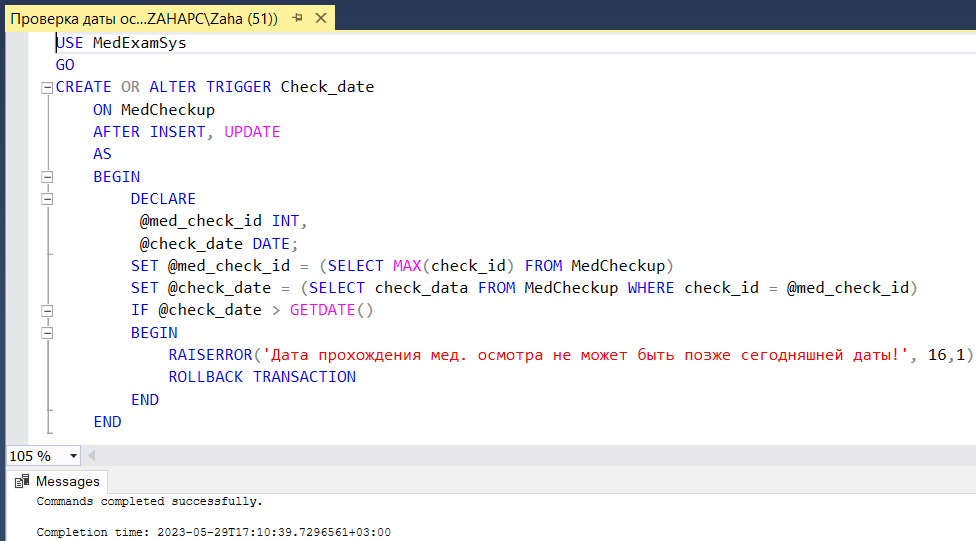


Рисунок 39 - Создание триггера Check\_date

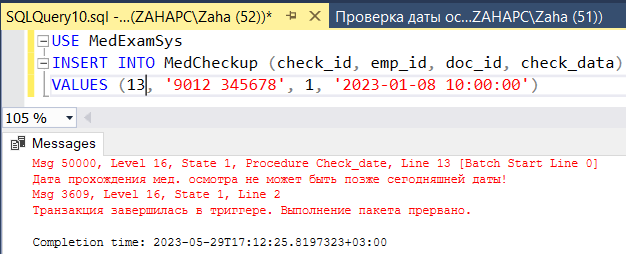


Рисунок 40 - Пример выполнения триггера Check\_date

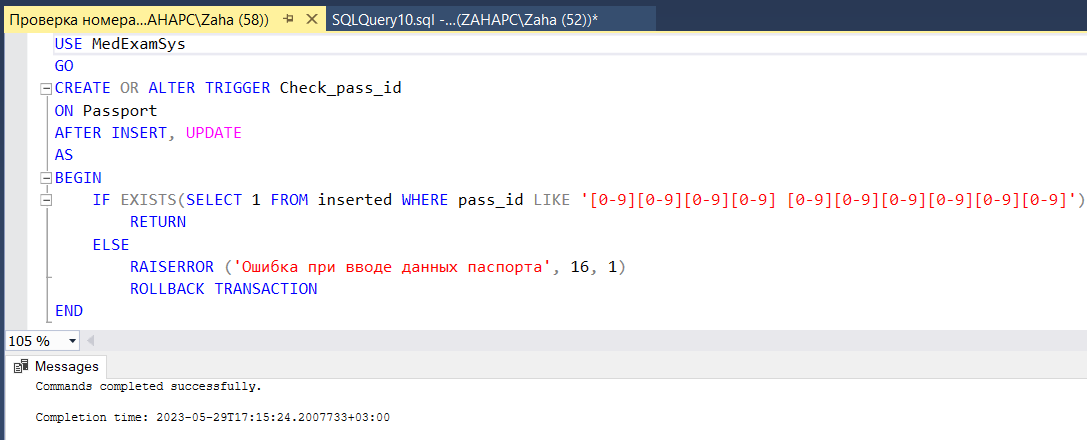


Рисунок 41 - Создание триггера Check\_pass\_id

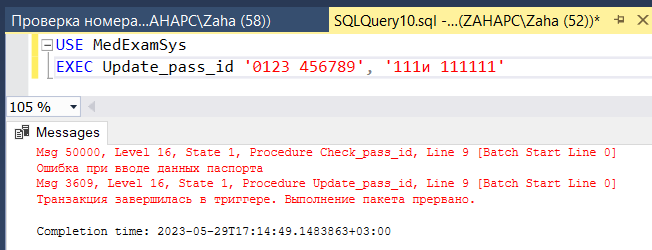


Рисунок 42 - Пример выполнения триггера Check\_pass\_id



Рисунок 43 - Создание триггера Check\_phone

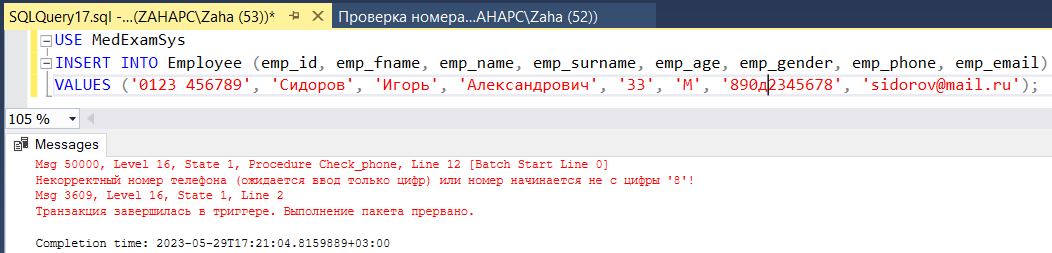


Рисунок 44 - Пример выполнения триггера Check\_phone

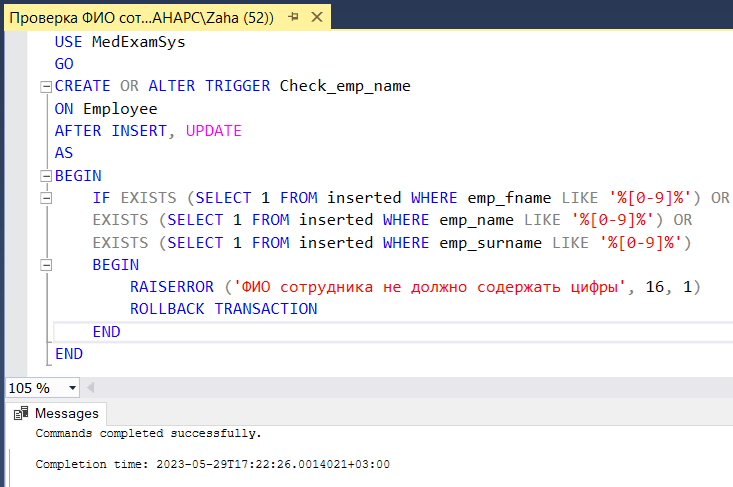


Рисунок 45 - Создание триггера Check\_emp\_name

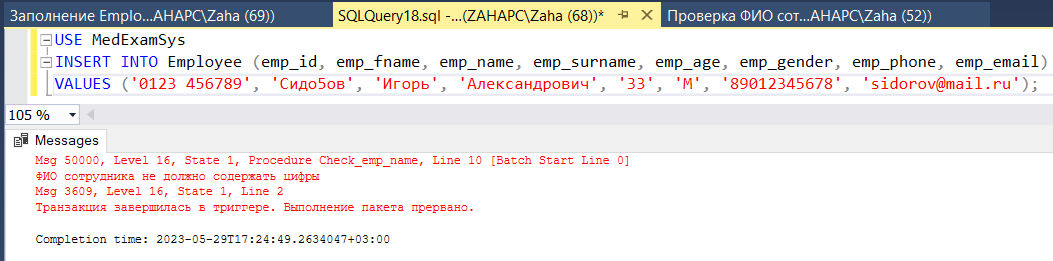


Рисунок 46 - Пример выполнения триггера Check\_emp\_name

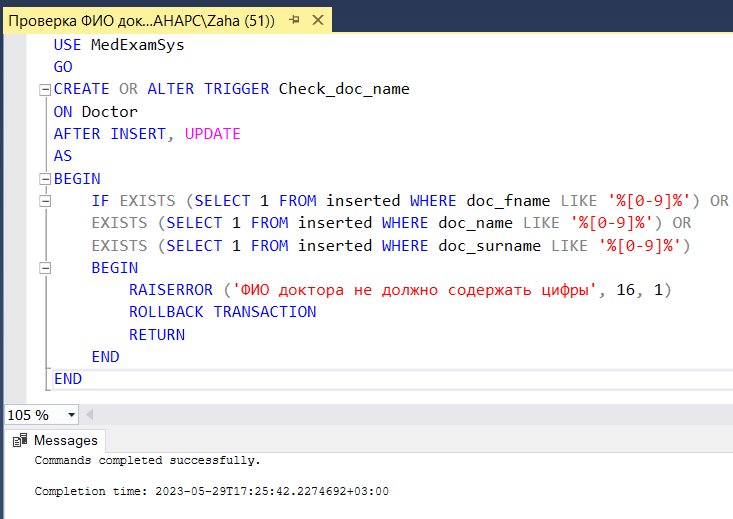


Рисунок 47 - Создание триггера Check\_doc\_name

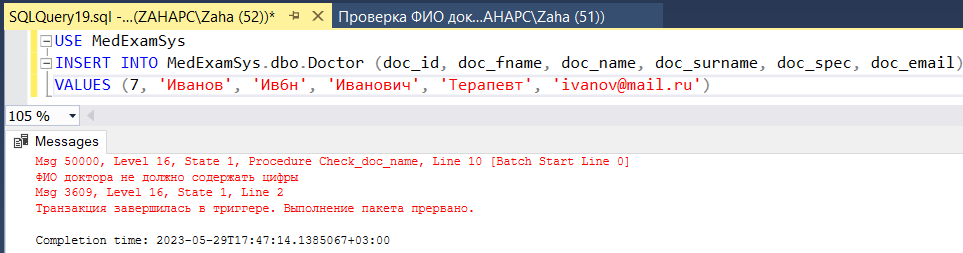


Рисунок 48 - Пример выполнения триггера Check\_doc\_name

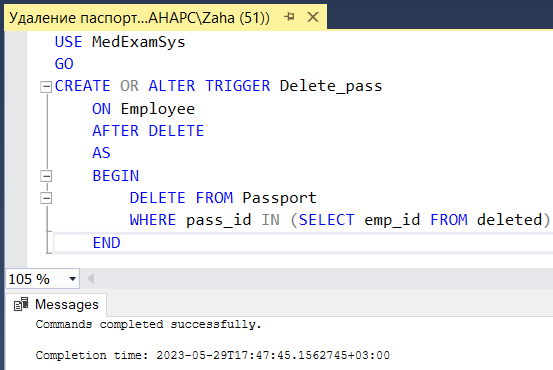


Рисунок 49 - Создание триггера Delete\_pass

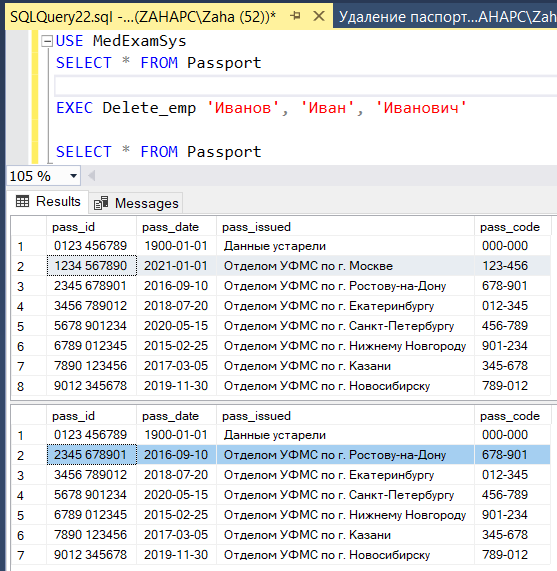


Рисунок 50 - Пример выполнения триггера Delete\_pass

Рисунки к главе 2:

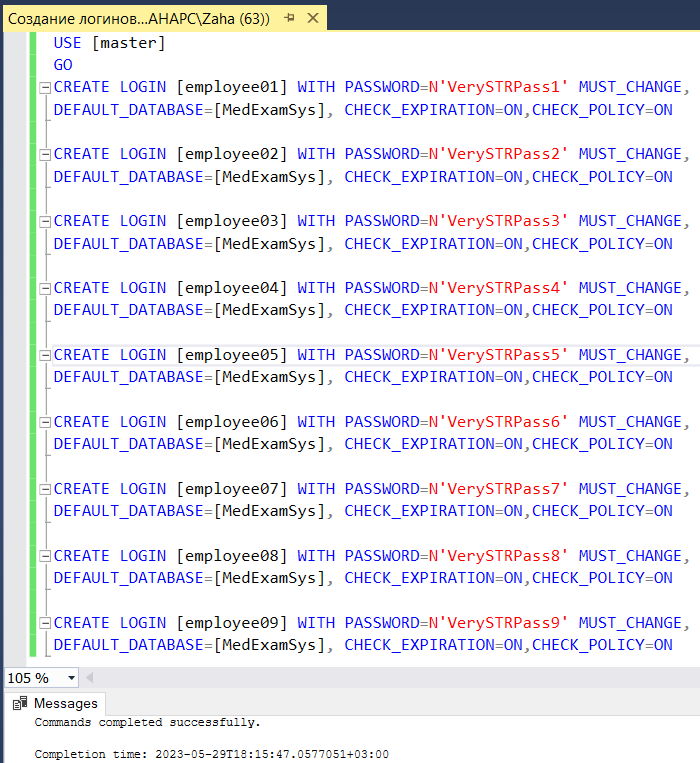


Рисунок 51 - Создание регистрационных имен

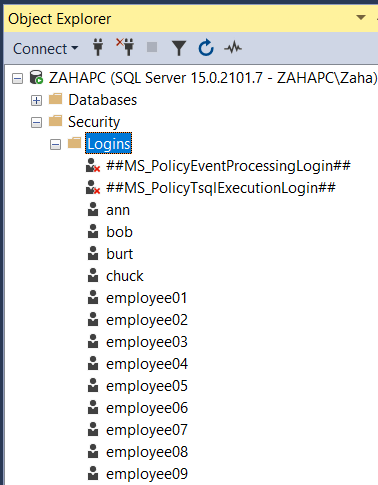


Рисунок 52 - Проверка отображения регистрационных имен

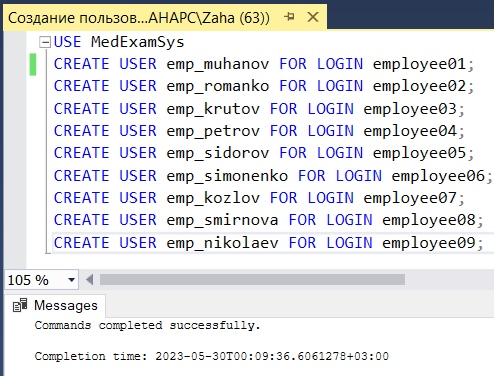


Рисунок 53 - Создание имен пользователей

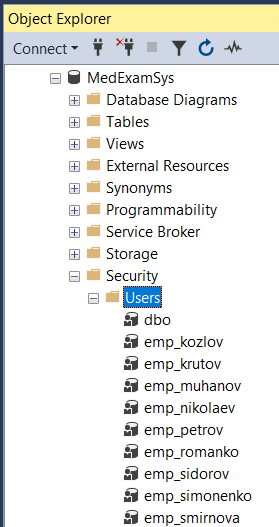


Рисунок 54 - Проверка отображения имен пользователей в Обозревателе объектов

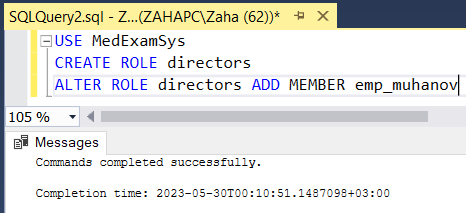


Рисунок 55 - Создание роли directors

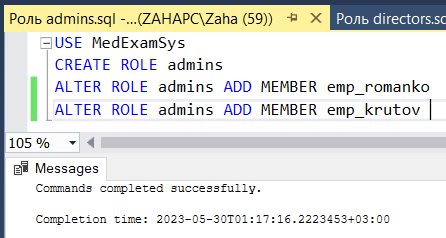


Рисунок 56 - Создание роли admins

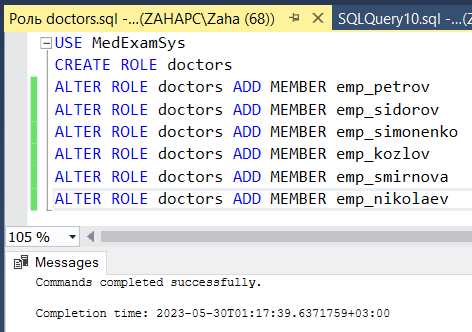


Рисунок 57 - Создание роли doctors

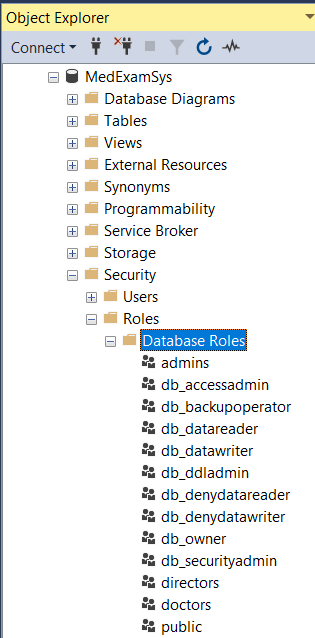


Рисунок 58 - Проверка отображения ролей в Обозревателе объектов

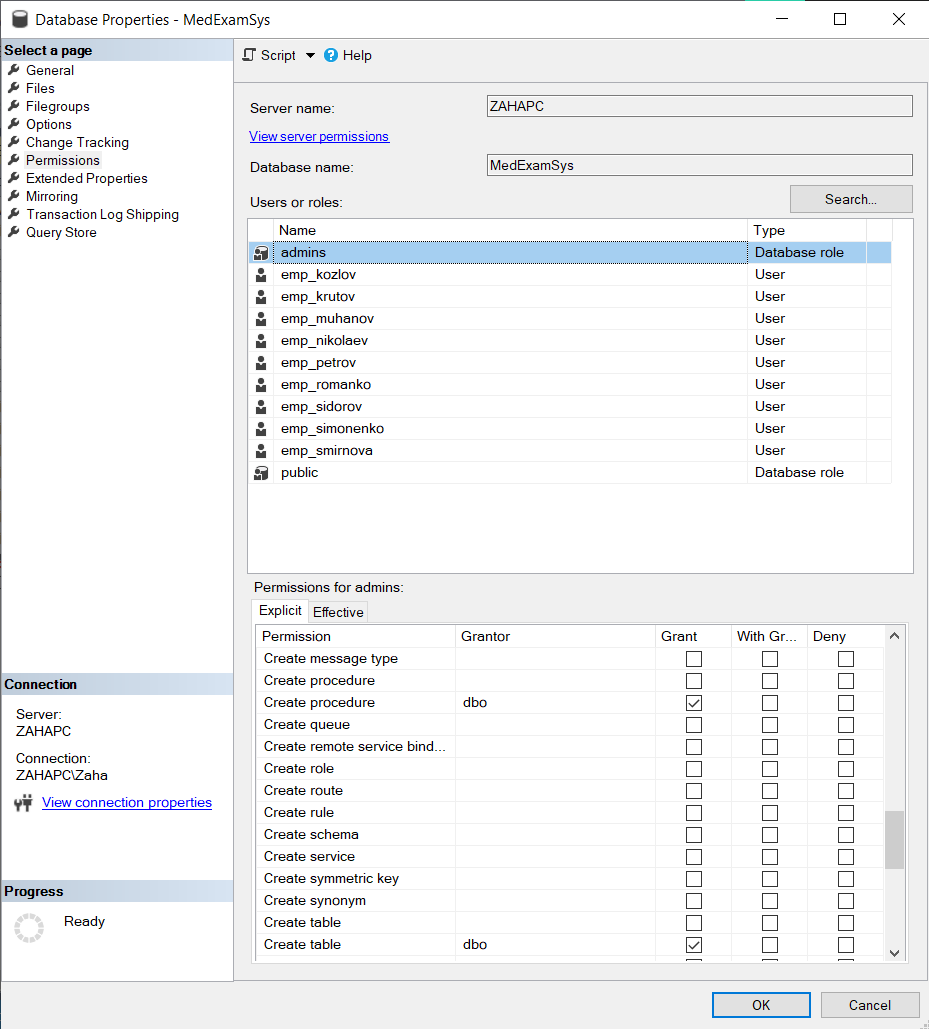


Рисунок 59 - Проверка разрешений роли admins в свойствах базы данных

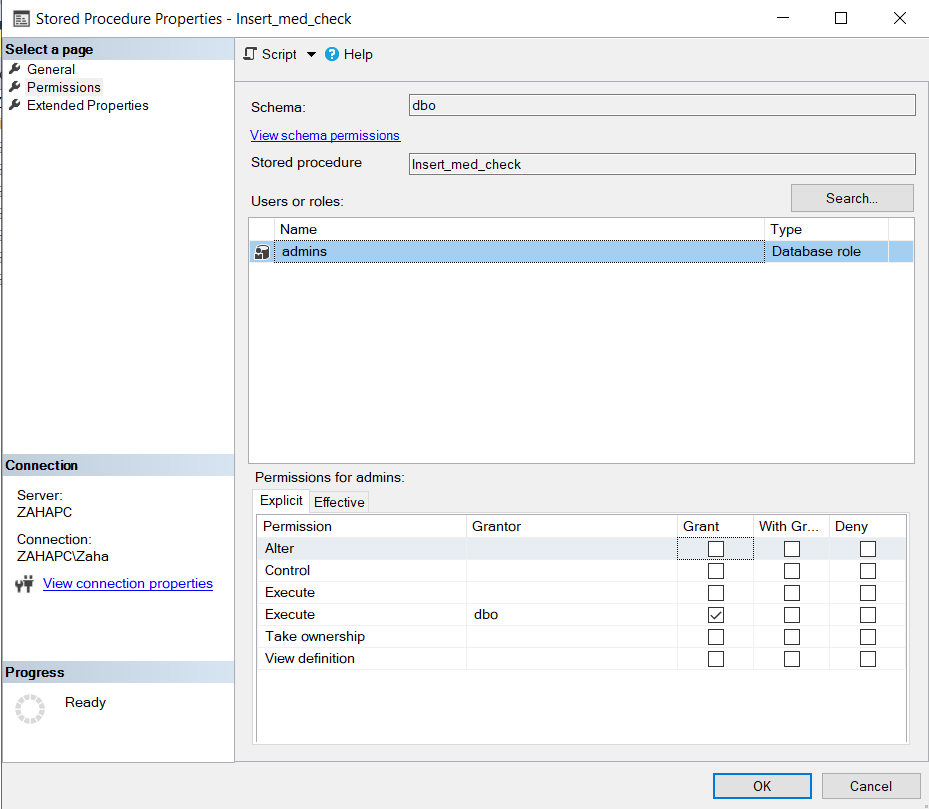


Рисунок 60 - Проверка разрешений роли admins в свойствах хранимой процедуры Insert\_med\_check

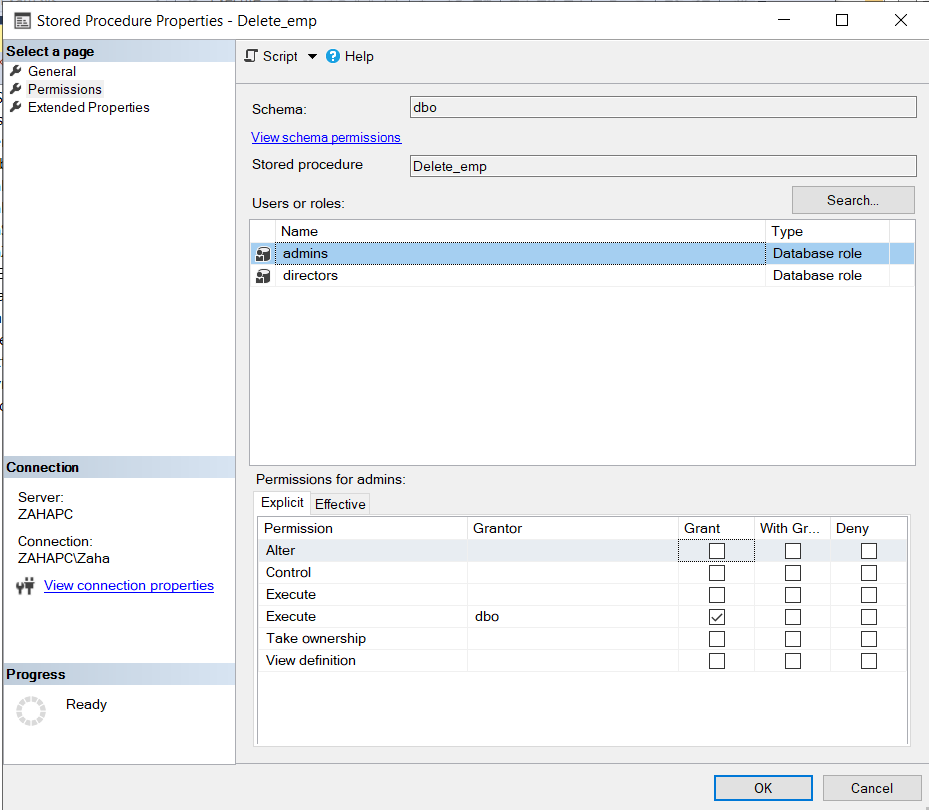


Рисунок 61 - Проверка разрешений роли admins в свойствах хранимой процедуры Delete\_emp

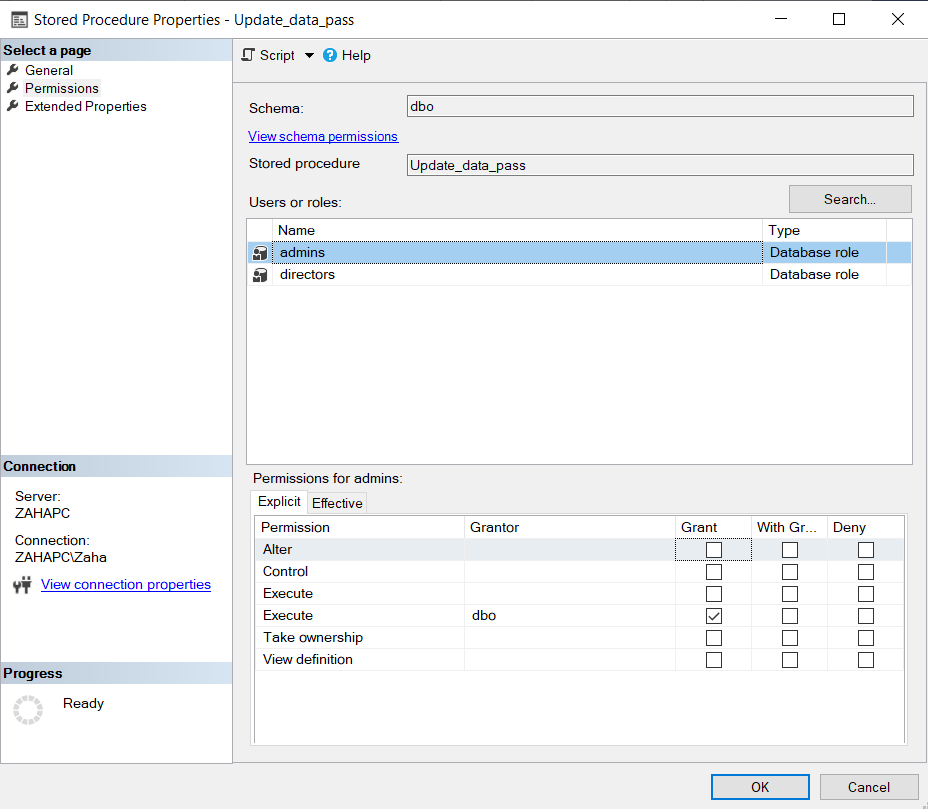


Рисунок 62 - Проверка разрешений роли admins в свойствах хранимой процедуры Update\_data\_pass

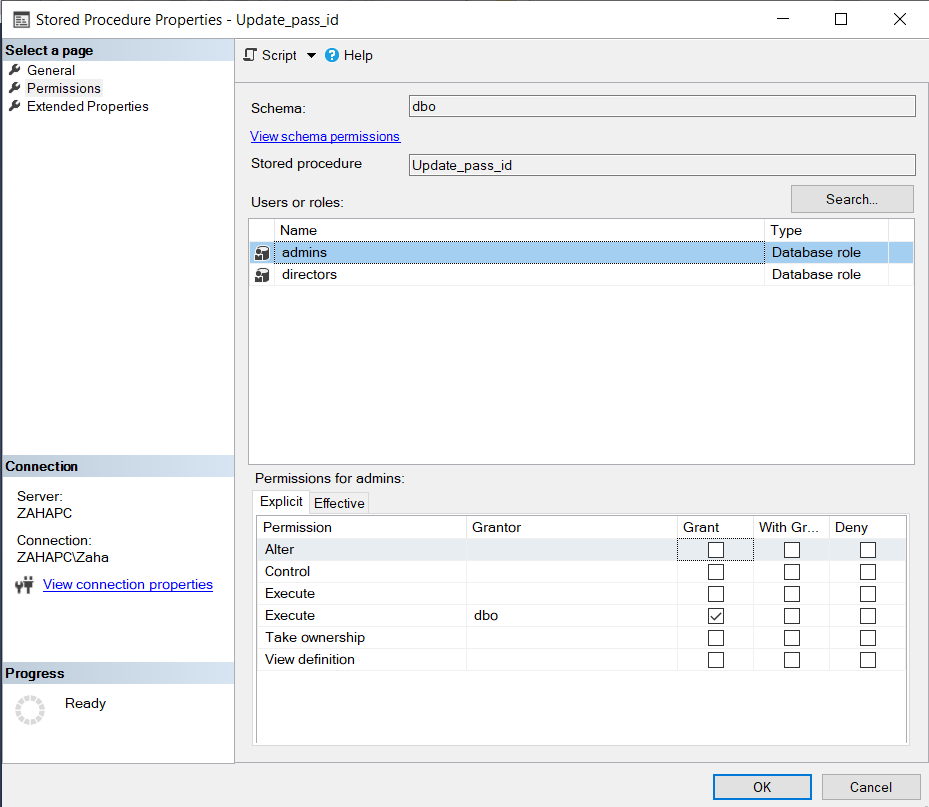


Рисунок 63 - Проверка разрешений роли admins в свойствах хранимой процедуры Update\_pass\_id

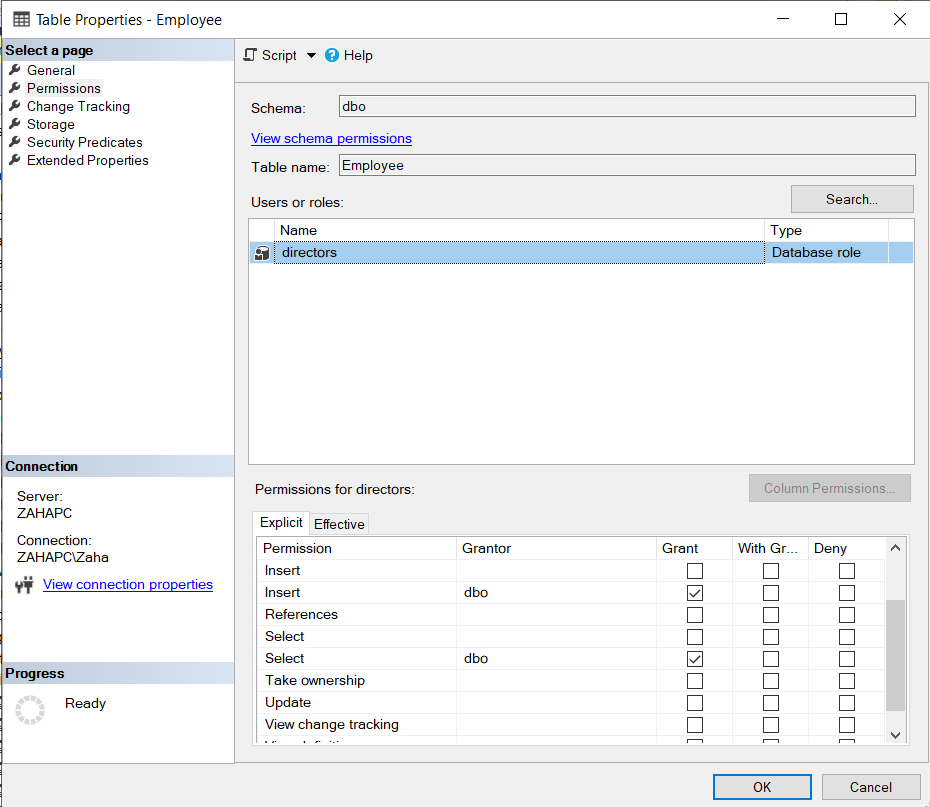


Рисунок 64 - Проверка разрешений роли directors в свойствах таблицы Employee

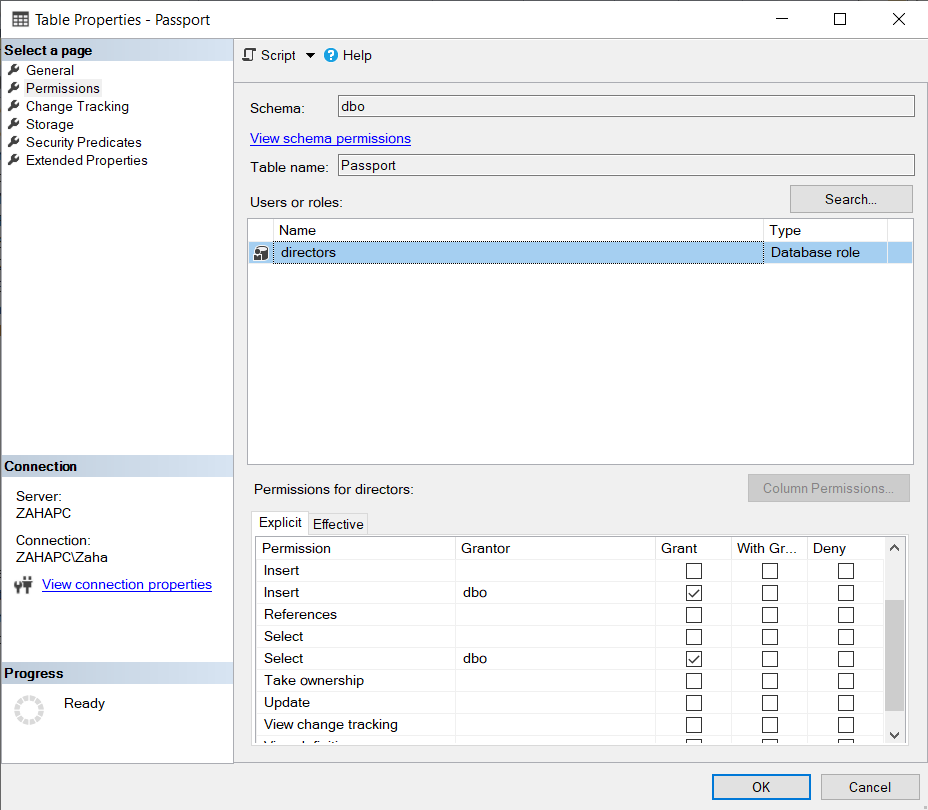


Рисунок 65 - Проверка разрешений роли directors в свойствах таблицы Passport

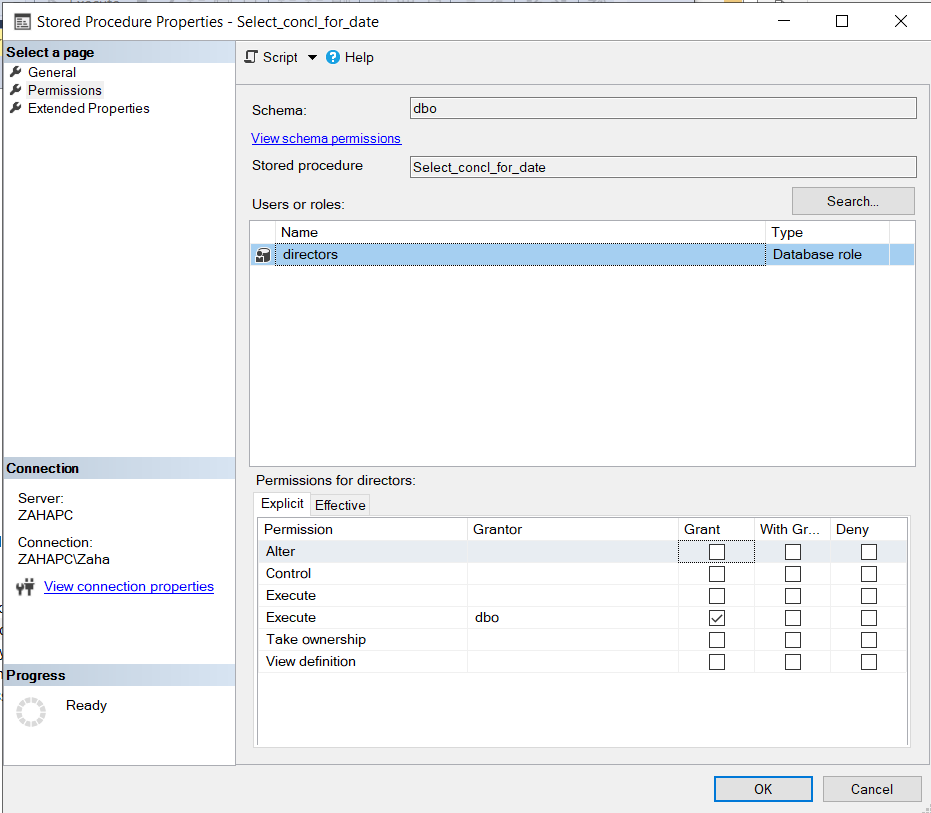


Рисунок 66 - Проверка разрешений роли directors в свойствах хранимой процедуры Select\_concl\_for\_date

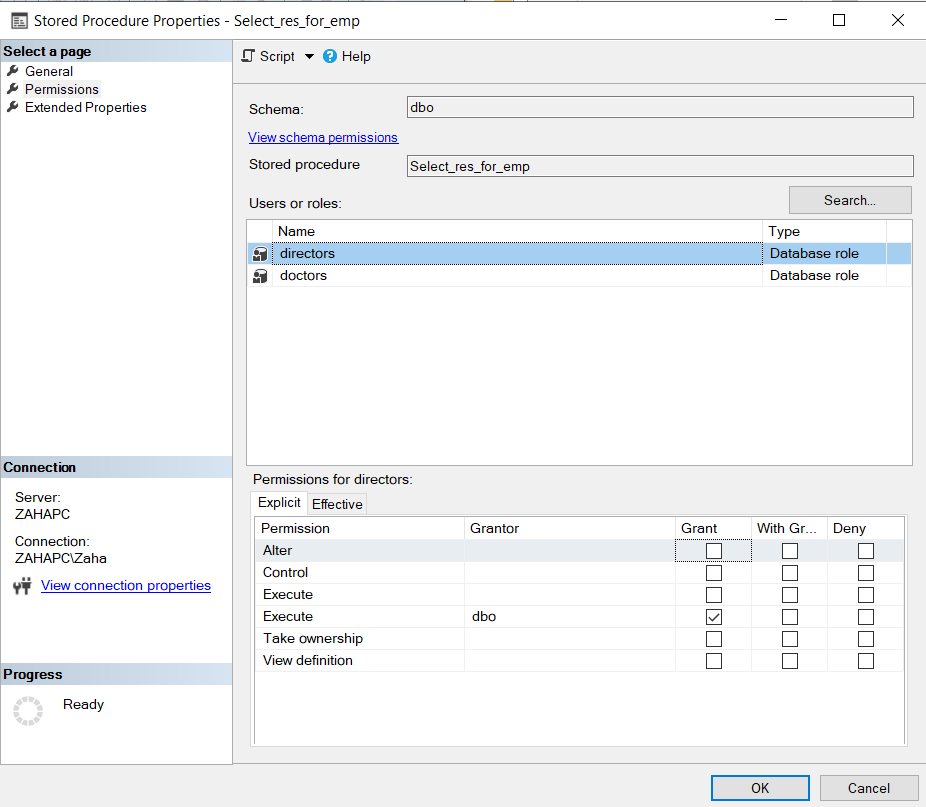


Рисунок 67 - Проверка разрешений роли directors в свойствах хранимой процедуры Select\_res\_for\_emp

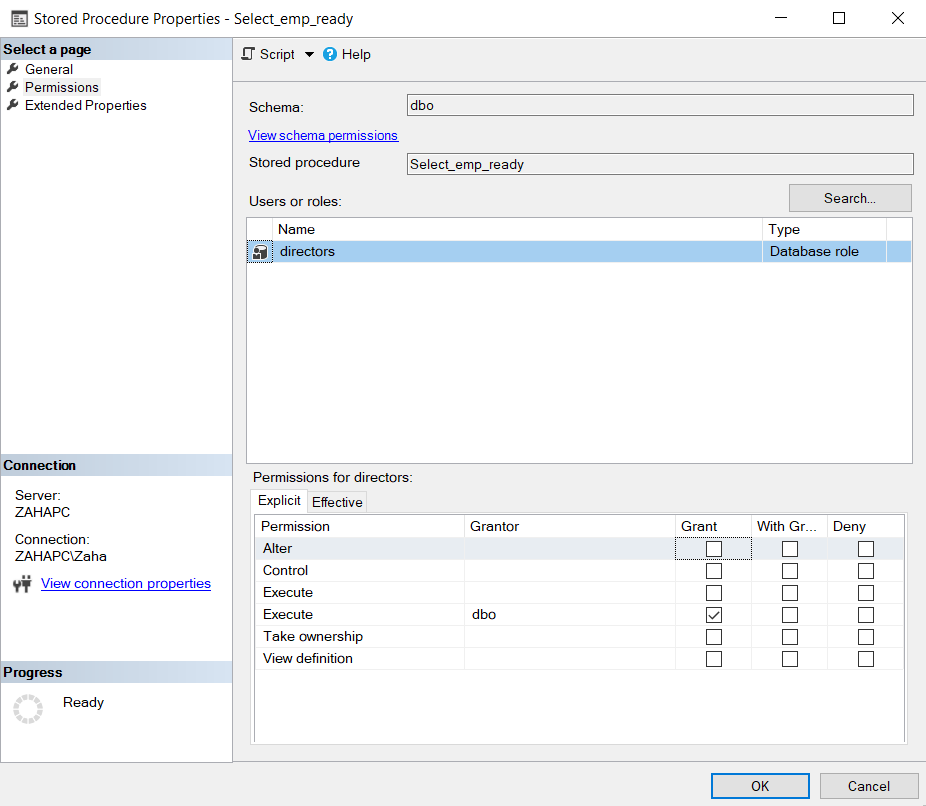


Рисунок 68 - Проверка разрешений роли directors в свойствах хранимой процедуры Select\_emp\_ready

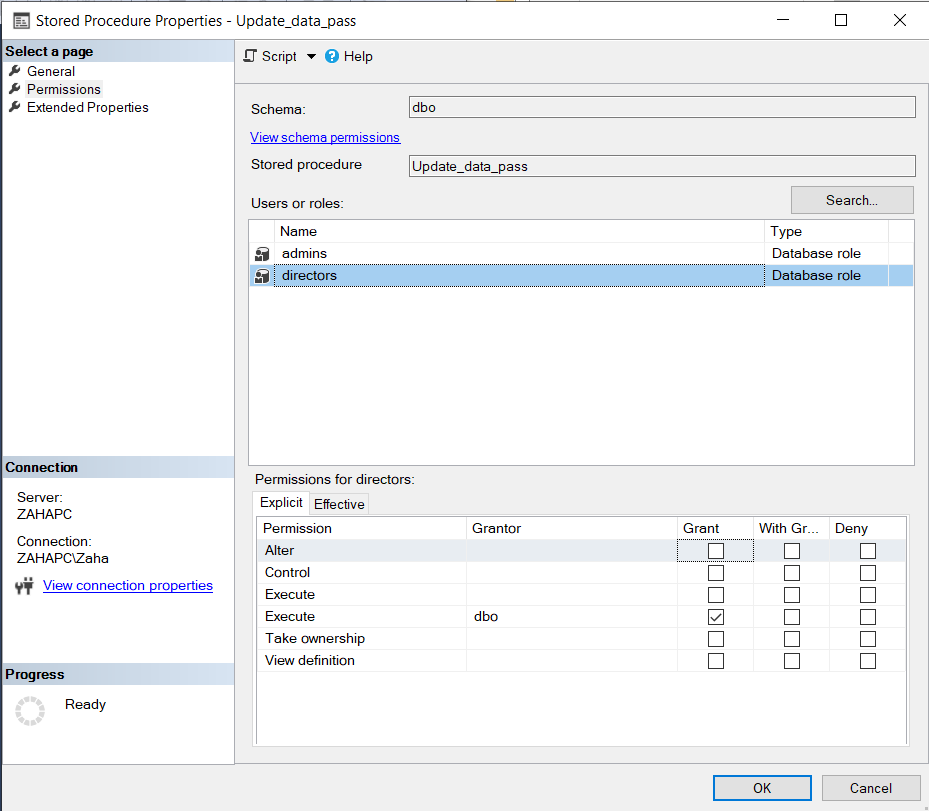


Рисунок 69 - Проверка разрешений роли directors в свойствах хранимой процедуры Update\_data\_pass

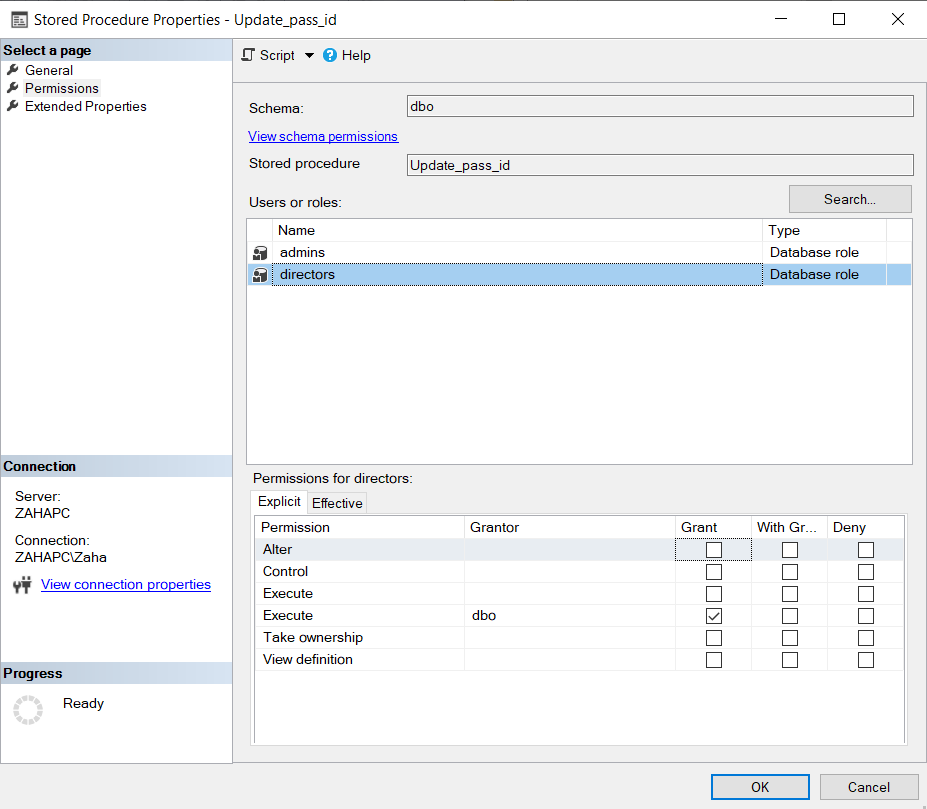


Рисунок 70 - Проверка разрешений роли directors в свойствах хранимой процедуры Update\_pass\_id

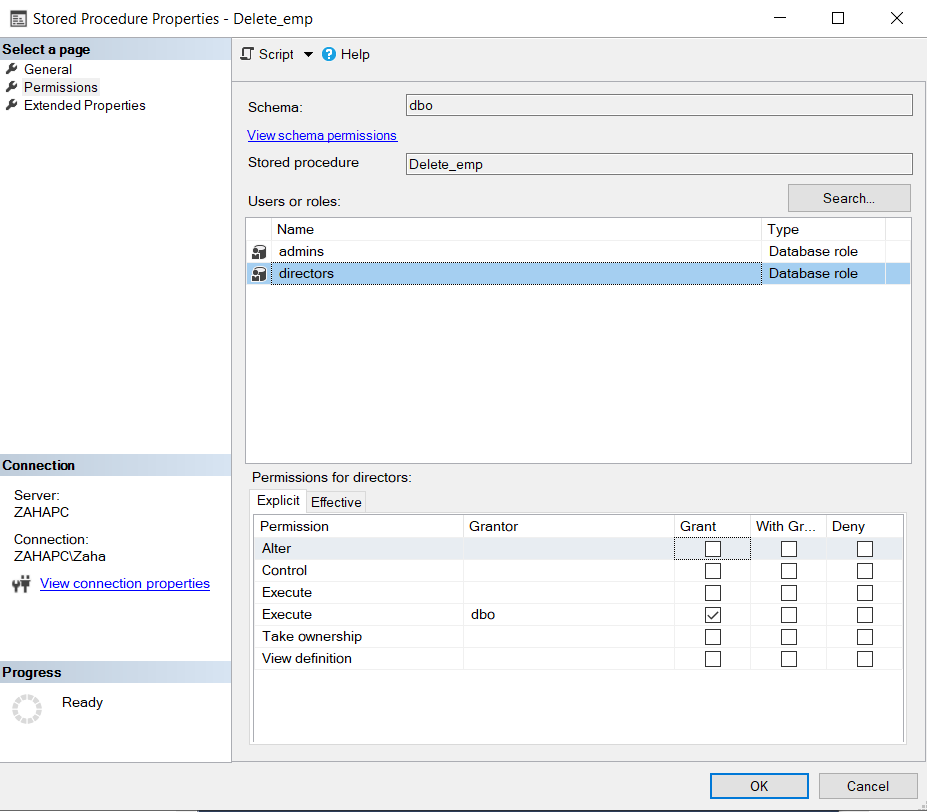


Рисунок 71 - Проверка разрешений роли directors в свойствах хранимой процедуры Delete\_emp

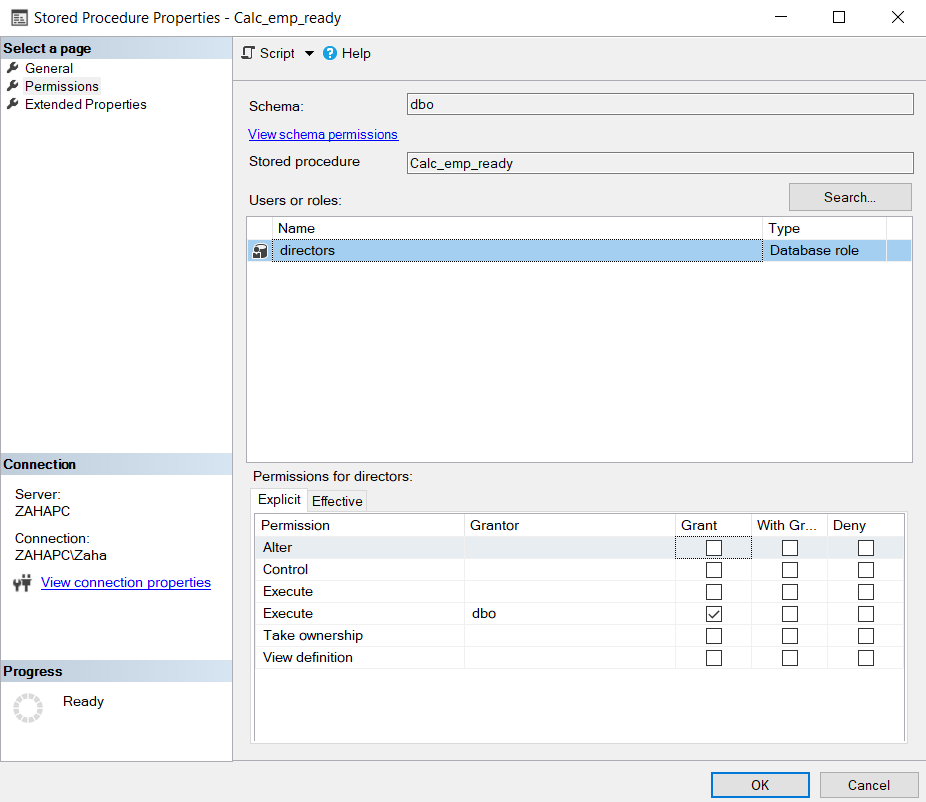


Рисунок 72 - Проверка разрешений роли directors в свойствах хранимой процедуры Calc\_emp\_ready

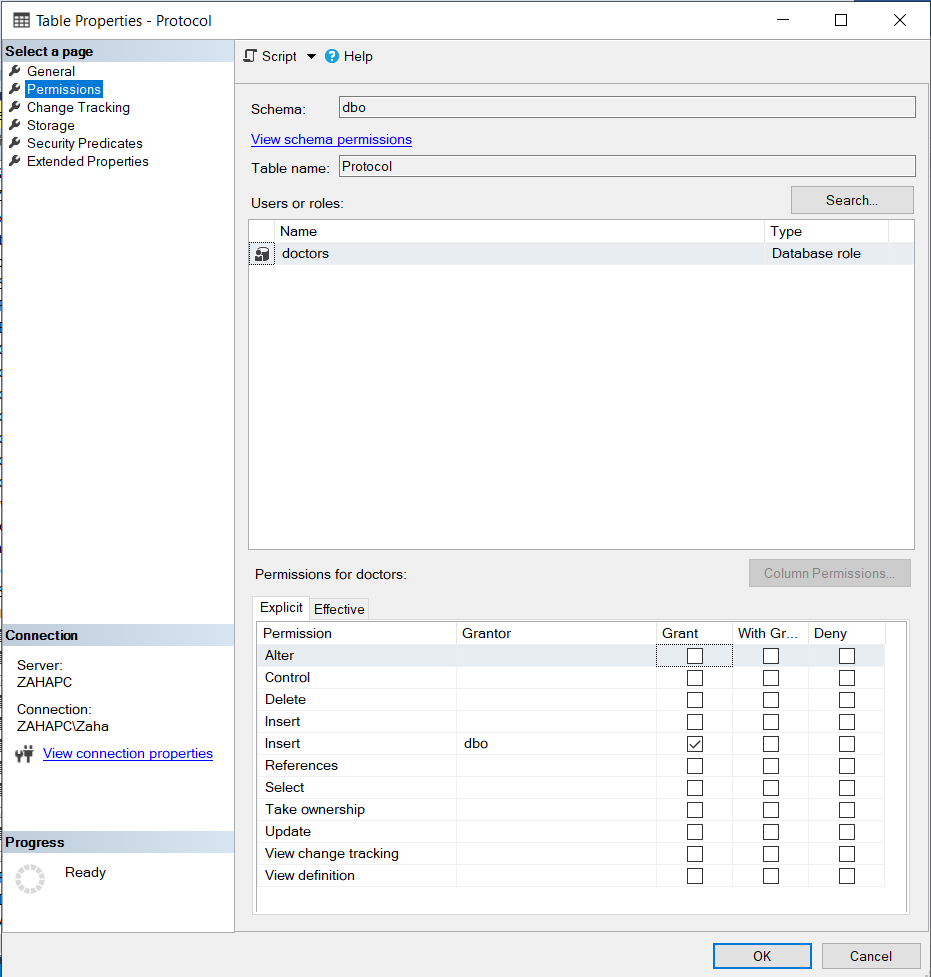


Рисунок 73 - Проверка разрешений роли doctors в свойствах таблицы Protocol

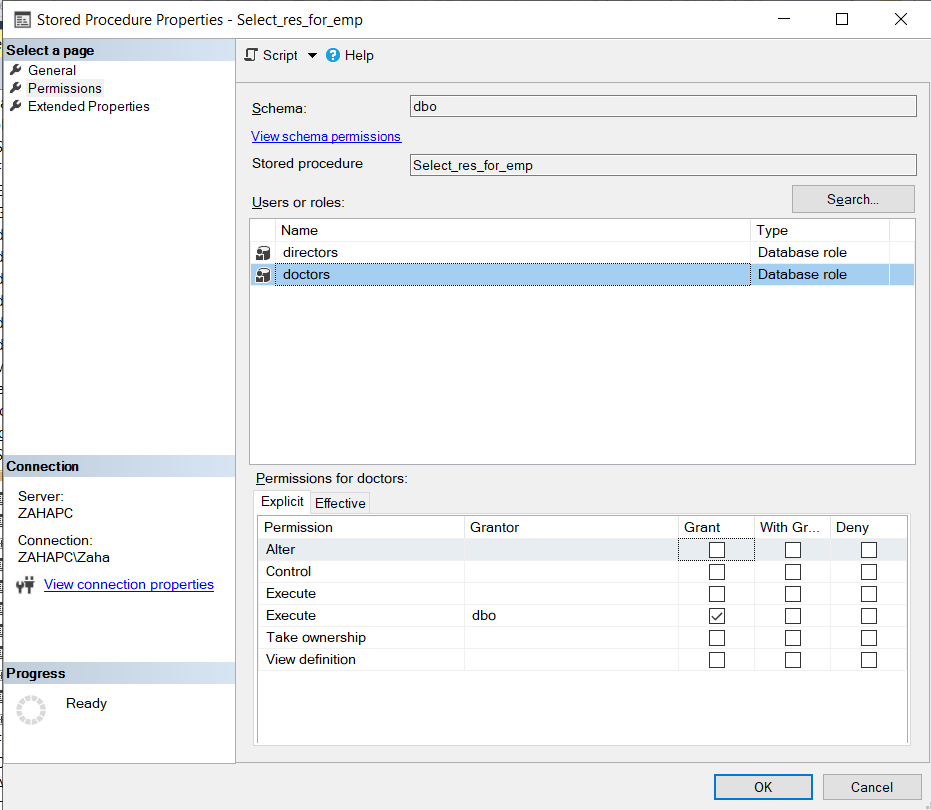


Рисунок 74 - Проверка разрешений роли directors в свойствах хранимой процедуры Select\_res\_for\_emp

Листинг создания базы данных:

USE MedExamSys;

CREATE TABLE Passport

(

pass\_id NCHAR(11) NOT NULL,

pass\_date DATE NOT NULL,

pass\_issued VARCHAR NOT NULL,

pass\_code NCHAR(7) NOT NULL,

CONSTRAINT prim\_pass\_id PRIMARY KEY(pass\_id)

)

CREATE TABLE Employee

(

emp\_id NCHAR(11) NOT NULL CONSTRAINT prim\_emp\_id PRIMARY KEY,

emp\_fname CHAR(30) NOT NULL,

emp\_name CHAR(30) NOT NULL,

emp\_surname CHAR(30) NULL,

emp\_age CHAR(3) NOT NULL,

emp\_gender CHAR(1) NOT NULL,

emp\_phone NCHAR(11) NOT NULL,

emp\_email NCHAR(20) NULL,

CONSTRAINT foreign\_pass FOREIGN KEY(emp\_id) REFERENCES Passport(pass\_id)

ON UPDATE CASCADE ON DELETE CASCADE

);CREATE INDEX emp\_name\_idx ON Employee(emp\_fname, emp\_name)

CREATE TABLE Doctor

(

doc\_id INT NOT NULL CONSTRAINT prim\_doc\_id PRIMARY KEY,

doc\_fname CHAR(30) NOT NULL,

doc\_name CHAR(30) NOT NULL,

doc\_surname CHAR(30) NULL,

doc\_spec CHAR(20) NOT NULL,

doc\_email NCHAR(20) NULL

);

CREATE INDEX doc\_name\_idx ON Doctor(doc\_fname, doc\_name)

CREATE TABLE MedCheckup

(

check\_id INT NOT NULL CONSTRAINT prim\_check\_id PRIMARY KEY,

emp\_id NCHAR(11) NOT NULL,

doc\_id INT NOT NULL,

check\_data DATETIME NOT NULL,

CONSTRAINT foreign\_emp FOREIGN KEY(emp\_id) REFERENCES Employee(emp\_id)

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT foreign\_doc FOREIGN KEY(doc\_id) REFERENCES Doctor(doc\_id)

CREATE TABLE Results

(

res\_id INT NOT NULL CONSTRAINT prim\_res\_id PRIMARY KEY,

res\_temp FLOAT NOT NULL,

res\_sys\_pres CHAR(3) NOT NULL,

res\_dias\_pres CHAR(3) NOT NULL,

res\_pulse CHAR(3) NOT NULL,

res\_alco FLOAT NOT NULL,

res\_compl NCHAR(100) NULL,

CONSTRAINT foreign\_check FOREIGN KEY(res\_id) REFERENCES MedCheckup(check\_id)

ON DELETE CASCADE

CREATE TABLE Protocol

(

prot\_num INT NOT NULL CONSTRAINT prim\_prot PRIMARY KEY,

prot\_concl NCHAR(100) NOT NULL,

CONSTRAINT foreign\_res FOREIGN KEY(prot\_num) REFERENCES Results(res\_id)

)

GO

CREATE or ALTER PROCEDURE Select\_concl\_for\_date

@med\_date DATE

WITH EXECUTE AS OWNER

AS

BEGIN

SELECT emp\_fname, emp\_name, emp\_surname, prot\_concl, check\_data

FROM Employee as emp, MedCheckup as med\_check, Protocol as prot

WHERE CAST(med\_check.check\_data AS DATE) = @med\_date

AND emp.emp\_id = med\_check.emp\_id AND

med\_check.check\_id = prot.prot\_num

END

GO

GO

CREATE or ALTER PROCEDURE Select\_res\_for\_emp

@emp\_fname CHAR(30),

@emp\_name CHAR(30),

@emp\_surname CHAR(30),

@flag BIT = 0

WITH EXECUTE AS OWNER

AS

BEGIN

IF @flag = 0

SELECT check\_id, check\_data, res\_temp, res\_sys\_pres, res\_dias\_pres, res\_pulse, res\_alco, res\_compl

FROM MedCheckup AS med\_check, Results AS res, Employee AS emp

WHERE emp.emp\_fname = @emp\_fname AND emp.emp\_name = @emp\_name AND

emp.emp\_surname = @emp\_surname AND emp.emp\_id = med\_check.emp\_id AND

med\_check.check\_id = res.res\_id

ELSE

SELECT check\_id, check\_data, res\_temp, res\_sys\_pres, res\_dias\_pres, res\_pulse, res\_alco, res\_compl, prot\_concl

FROM MedCheckup AS med\_check, Results AS res, Employee AS emp, Protocol AS prot

WHERE emp.emp\_fname = @emp\_fname AND emp.emp\_name = @emp\_name AND

emp.emp\_surname = @emp\_surname AND emp.emp\_id = med\_check.emp\_id AND

med\_check.check\_id = res.res\_id AND res.res\_id = prot.prot\_num

END

GO

GO

CREATE or ALTER PROCEDURE Select\_emp\_ready

WITH EXECUTE AS OWNER

AS

BEGIN

SELECT emp\_fname, emp\_name, emp\_surname, check\_data

FROM MedCheckup AS med\_check, Protocol AS prot, Employee AS emp

WHERE emp.emp\_id = med\_check.emp\_id AND

med\_check.check\_id = prot\_num AND

prot\_concl = 'Признаки состояний и заболеваний отсутствуют'

END

GO

GO

CREATE OR ALTER PROCEDURE Insert\_med\_check

@new\_check\_id INT,

@emp\_id NCHAR(11),

@doc\_id INT,

@new\_check\_date DATETIME

WITH EXECUTE AS OWNER

AS

BEGIN

IF EXISTS(SELECT \* FROM MedCheckup WHERE check\_id = @new\_check\_id)

UPDATE MedCheckup

SET emp\_id = @emp\_id,

doc\_id = @doc\_id,

check\_data = @new\_check\_date

WHERE check\_id = @new\_check\_id

ELSE

INSERT INTO MedCheckup

VALUES (@new\_check\_id, @emp\_id, @doc\_id, @new\_check\_date)

END

GO

CREATE OR ALTER PROCEDURE Update\_data\_pass

@new\_pass\_id NCHAR(11),

@new\_pass\_date DATE,

@new\_pass\_issued VARCHAR(100),

@new\_pass\_code NCHAR(7)

WITH EXECUTE AS OWNER

AS

DISABLE TRIGGER Update\_old\_pass ON Passport

BEGIN

IF EXISTS(SELECT pass\_id FROM Passport WHERE pass\_id = @new\_pass\_id)

BEGIN

UPDATE Passport

SET pass\_date = @new\_pass\_date,

pass\_issued = @new\_pass\_issued,

pass\_code = @new\_pass\_code

WHERE pass\_id = @new\_pass\_id

END

ELSE

RAISERROR ('Таких номера и серии паспорта не существует для обновления', 16, 1)

END

GO

CREATE OR ALTER PROCEDURE Update\_pass\_id

@old\_pass\_id NCHAR(11),

@new\_pass\_id NCHAR(11)

WITH EXECUTE AS OWNER

AS

ENABLE TRIGGER Update\_old\_pass ON Passport

IF NOT EXISTS (SELECT pass\_id FROM Passport WHERE pass\_id = @new\_pass\_id)

BEGIN

UPDATE Passport SET pass\_id = @new\_pass\_id

WHERE pass\_id = @old\_pass\_id

END

ELSE

RAISERROR ('Такие номер и серия паспорта уже существуют в базе данных', 16, 1)

GO

CREATE OR ALTER PROCEDURE Calc\_emp\_ready

WITH EXECUTE AS OWNER

AS

BEGIN

SELECT COUNT(DISTINCT emp\_id) AS count\_ready

FROM MedCheckup AS m, Protocol AS p

WHERE m.check\_id = p.prot\_num AND

p.prot\_concl = 'Признаки состояний и заболеваний отсутствуют'

END

GO

CREATE OR ALTER PROCEDURE Delete\_emp

@emp\_fname CHAR(30),

@emp\_name CHAR(30),

@emp\_surname CHAR(30)

WITH EXECUTE AS OWNER

AS

BEGIN

DELETE FROM Employee

WHERE emp\_fname = @emp\_fname AND

emp\_name = @emp\_name AND

emp\_surname = @emp\_surname

END

GO

CREATE OR ALTER TRIGGER Update\_old\_pass

ON Passport

AFTER UPDATE

AS

BEGIN

UPDATE Passport

SET pass\_code = '000-000',

pass\_date = '1900-01-01',

pass\_issued = 'Данные устарели'

WHERE pass\_id = (SELECT pass\_id FROM inserted)

END

GO

CREATE OR ALTER TRIGGER Insert\_prot\_concl

ON Protocol

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @new\_prot\_id INT,

@emp\_fname CHAR(30),

@emp\_name CHAR(30),

@emp\_surname CHAR(30)

SELECT @new\_prot\_id = inserted.prot\_num FROM inserted

SELECT @emp\_fname = (SELECT emp\_fname FROM Employee AS e, MedCheckup AS m

WHERE e.emp\_id = m.emp\_id AND m.check\_id = @new\_prot\_id)

SELECT @emp\_name = (SELECT emp\_name FROM Employee AS e, MedCheckup AS m

WHERE e.emp\_id = m.emp\_id AND m.check\_id = @new\_prot\_id)

SELECT @emp\_surname = (SELECT emp\_surname FROM Employee AS e, MedCheckup AS m

WHERE e.emp\_id = m.emp\_id AND m.check\_id = @new\_prot\_id)

EXEC Select\_res\_for\_emp @emp\_fname, @emp\_name, @emp\_surname, 1

END

GO

CREATE OR ALTER TRIGGER Check\_date

ON MedCheckup

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE

@med\_check\_id INT,

@check\_date DATE;

SET @med\_check\_id = (SELECT MAX(check\_id) FROM MedCheckup)

SET @check\_date = (SELECT check\_data FROM MedCheckup WHERE check\_id = @med\_check\_id)

IF @check\_date > GETDATE()

BEGIN

RAISERROR('Дата прохождения мед. осмотра не может быть позже сегодняшней даты!', 16,1)

ROLLBACK TRANSACTION

END

END

GO

CREATE OR ALTER TRIGGER Check\_pass\_id

ON Passport

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS(SELECT 1 FROM inserted WHERE pass\_id LIKE '[0-9][0-9][0-9][0-9] [0-9][0-9][0-9][0-9][0-9][0-9]')

RETURN

ELSE

RAISERROR ('Ошибка при вводе данных паспорта', 16, 1)

ROLLBACK TRANSACTION

END

GO

CREATE OR ALTER TRIGGER Check\_phone

ON Employee

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @phone NCHAR(11)

SELECT @phone = inserted.emp\_phone FROM inserted

IF (@phone LIKE '8[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

RETURN

ELSE

BEGIN

RAISERROR('Некорректный номер телефона (ожидается ввод только цифр) или номер начинается не с цифры ''8''!', 16,1)

ROLLBACK TRANSACTION

END

END

GO

CREATE OR ALTER TRIGGER Check\_doc\_name

ON Doctor

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT 1 FROM inserted WHERE doc\_fname LIKE '%[0-9]%') OR

EXISTS (SELECT 1 FROM inserted WHERE doc\_name LIKE '%[0-9]%') OR

EXISTS (SELECT 1 FROM inserted WHERE doc\_surname LIKE '%[0-9]%')

BEGIN

RAISERROR ('ФИО доктора не должно содержать цифры', 16, 1)

ROLLBACK TRANSACTION

RETURN

END

END

GO

CREATE OR ALTER TRIGGER Check\_emp\_name

ON Employee

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT 1 FROM inserted WHERE emp\_fname LIKE '%[0-9]%') OR

EXISTS (SELECT 1 FROM inserted WHERE emp\_name LIKE '%[0-9]%') OR

EXISTS (SELECT 1 FROM inserted WHERE emp\_surname LIKE '%[0-9]%')

BEGIN

RAISERROR ('ФИО сотрудника не должно содержать цифры', 16, 1)

ROLLBACK TRANSACTION

END

END

GO

CREATE OR ALTER TRIGGER Delete\_pass

ON Employee

AFTER DELETE

AS

BEGIN

DELETE FROM Passport

WHERE pass\_id IN (SELECT emp\_id FROM deleted)

END

USE [master]

GO

CREATE LOGIN [employee01] WITH PASSWORD=N'VerySTRPass1' MUST\_CHANGE,

DEFAULT\_DATABASE=[MedExamSys], CHECK\_EXPIRATION=ON,CHECK\_POLICY=ON

CREATE LOGIN [employee02] WITH PASSWORD=N'VerySTRPass2' MUST\_CHANGE,

DEFAULT\_DATABASE=[MedExamSys], CHECK\_EXPIRATION=ON,CHECK\_POLICY=ON

CREATE LOGIN [employee03] WITH PASSWORD=N'VerySTRPass3' MUST\_CHANGE,

DEFAULT\_DATABASE=[MedExamSys], CHECK\_EXPIRATION=ON,CHECK\_POLICY=ON

CREATE LOGIN [employee04] WITH PASSWORD=N'VerySTRPass4' MUST\_CHANGE,

DEFAULT\_DATABASE=[MedExamSys], CHECK\_EXPIRATION=ON,CHECK\_POLICY=ON

CREATE LOGIN [employee05] WITH PASSWORD=N'VerySTRPass5' MUST\_CHANGE,

DEFAULT\_DATABASE=[MedExamSys], CHECK\_EXPIRATION=ON,CHECK\_POLICY=ON

CREATE LOGIN [employee06] WITH PASSWORD=N'VerySTRPass6' MUST\_CHANGE,

DEFAULT\_DATABASE=[MedExamSys], CHECK\_EXPIRATION=ON,CHECK\_POLICY=ON

CREATE LOGIN [employee07] WITH PASSWORD=N'VerySTRPass7' MUST\_CHANGE,

DEFAULT\_DATABASE=[MedExamSys], CHECK\_EXPIRATION=ON,CHECK\_POLICY=ON

CREATE LOGIN [employee08] WITH PASSWORD=N'VerySTRPass8' MUST\_CHANGE,

DEFAULT\_DATABASE=[MedExamSys], CHECK\_EXPIRATION=ON,CHECK\_POLICY=ON

CREATE LOGIN [employee09] WITH PASSWORD=N'VerySTRPass9' MUST\_CHANGE,

DEFAULT\_DATABASE=[MedExamSys], CHECK\_EXPIRATION=ON,CHECK\_POLICY=ON

USE MedExamSys

CREATE USER emp\_muhanov FOR LOGIN employee01;

CREATE USER emp\_romanko FOR LOGIN employee02;

CREATE USER emp\_krutov FOR LOGIN employee03;

CREATE USER emp\_petrov FOR LOGIN employee04;

CREATE USER emp\_sidorov FOR LOGIN employee05;

CREATE USER emp\_simonenko FOR LOGIN employee06;

CREATE USER emp\_kozlov FOR LOGIN employee07;

CREATE USER emp\_smirnova FOR LOGIN employee08;

CREATE USER emp\_nikolaev FOR LOGIN employee09;

USE MedExamSys

CREATE ROLE admins

ALTER ROLE admins ADD MEMBER emp\_romanko

ALTER ROLE admins ADD MEMBER emp\_krutov

USE MedExamSys

CREATE ROLE directors

ALTER ROLE directors ADD MEMBER emp\_muhanov

USE MedExamSys

CREATE ROLE doctors

ALTER ROLE doctors ADD MEMBER emp\_petrov

ALTER ROLE doctors ADD MEMBER emp\_sidorov

ALTER ROLE doctors ADD MEMBER emp\_simonenko

ALTER ROLE doctors ADD MEMBER emp\_kozlov

ALTER ROLE doctors ADD MEMBER emp\_smirnova

ALTER ROLE doctors ADD MEMBER emp\_nikolaev

USE MedExamSys

GRANT CREATE PROCEDURE TO admins

GRANT CREATE TABLE TO admins

GRANT EXECUTE ON Insert\_med\_check TO admins

GRANT EXECUTE ON Delete\_emp TO admins

GRANT EXECUTE ON Update\_pass\_id TO admins

GRANT EXECUTE ON Update\_data\_pass TO admins

USE MedExamSys

GRANT INSERT ON Passport TO directors

GRANT INSERT ON Employee TO directors

GRANT SELECT ON Passport TO directors

GRANT SELECT ON Employee TO directors

GRANT EXECUTE ON Select\_concl\_for\_date TO directors

GRANT EXECUTE ON Select\_res\_for\_emp TO directors

GRANT EXECUTE ON Select\_emp\_ready TO directors

GRANT EXECUTE ON Update\_data\_pass TO directors

GRANT EXECUTE ON Update\_pass\_id TO directors

GRANT EXECUTE ON Delete\_emp TO directors

GRANT EXECUTE ON Calc\_emp\_ready TO directors

USE MedExamSys

GRANT INSERT ON Protocol TO doctors

GRANT EXECUTE ON Select\_res\_for\_emp To doctors

USE MedExamSys

DENY SELECT ON Employee TO doctors

DENY SELECT ON Passport TO doctors

USE MedExamSys

REVOKE EXECUTE ON Delete\_emp TO directors

USE MedExamSys

CREATE MASTER KEY

ENCRYPTION BY PASSWORD = 'DMKbdMedExamSys2023'

USE MedExamSys

CREATE ASYMMETRIC KEY ASMKEY

WITH ALGORITHM = RSA\_2048

ENCRYPTION BY PASSWORD = 'ASYMKEYMedExamSys2023'

USE MedExamSys

CREATE SYMMETRIC KEY SYMKEY

WITH ALGORITHM = AES\_256

ENCRYPTION BY ASYMMETRIC KEY ASMKEY

USE MedExamSys

OPEN SYMMETRIC KEY SYMKEY

DECRYPTION BY ASYMMETRIC KEY ASMKEY

WITH PASSWORD = 'ASYMKEYMedExamSys2023'

USE MedExamSys

ALTER TABLE Employee

ADD emp\_enc\_phone VARBINARY(100)

GO

UPDATE Employee

SET emp\_enc\_phone = ENCRYPTBYKEY(KEY\_GUID('SYMKEY'), emp\_phone)

GO

SELECT emp\_phone, emp\_enc\_phone,

CONVERT(NCHAR(11), DECRYPTBYKEY(emp\_enc\_phone)) AS emp\_dec\_phone

FROM Employee

USE master;

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MedExamSysMASTER'

BACKUP MASTER KEY

TO FILE = 'D:\MSS\MSSQL15.MSSQLSERVER\MSSQL\Backup\Master.bak'

ENCRYPTION BY PASSWORD = 'MedExamSysBACKUP'

GO

SELECT \* FROM sys.key\_encryptions

USE master;

CREATE CERTIFICATE MedExamSysCERT WITH SUBJECT = 'BDMESCERT'

BACKUP CERTIFICATE MedExamSysCERT

TO FILE = 'D:\MSS\MSSQL15.MSSQLSERVER\MSSQL\Backup\CERT.bak'

WITH PRIVATE KEY

(FILE = 'D:\MSS\MSSQL15.MSSQLSERVER\MSSQL\Backup\PRIVATE.bak',

ENCRYPTION BY PASSWORD = 'MESprivCRT')

GO

SELECT \* FROM sys.certificates WHERE name = 'MedExamSysCERT'

USE MedExamSys;

CREATE DATABASE ENCRYPTION KEY

WITH ALGORITHM = AES\_128

ENCRYPTION BY SERVER CERTIFICATE MedExamSysCERT;

ALTER DATABASE MedExamSys SET ENCRYPTION ON;

SELECT DB\_NAME(database\_id), \*

FROM sys.dm\_database\_encryption\_keys